mogasens_csv

Generated by Doxygen 1.8.17

1 Namespace Index 1
1.1 Namespace List . . . 1

2 Hierarchical Index 3
2.1 Class Hierarchy . . . 3

3 Class Index 5
3.1 Class List . . . 5

4 File Index 7
4.1 File List . . . 7

5 Namespace Documentation 11
5.1 cl Namespace Reference . . . 11
5.1.1 Typedef Documentation . . . 13
5.1.1.1 column_type . . . 13
5.1.1.2 Expected . . . 13
5.1.2 Enumeration Type Documentation . . . 13
5.1.2.1 Channel . . . 13
5.1.2.2 Column . . . 14
5.1.2.3 CsvFileKind . . . 14
5.1.2.4 Sensor . . . 14
5.1.3 Function Documentation . . . 15
5.1.3.1 CL_SPECIALIZE_COL_TRAITS() [1/11] . . . 15
5.1.3.2 CL_SPECIALIZE_COL_TRAITS() [2/11] . . . 15
5.1.3.3 CL_SPECIALIZE_COL_TRAITS() [3/11] . . . 15
5.1.3.4 CL_SPECIALIZE_COL_TRAITS() [4/11] . . . 15
5.1.3.5 CL_SPECIALIZE_COL_TRAITS() [5/11] . . . 15
5.1.3.6 CL_SPECIALIZE_COL_TRAITS() [6/11] . . . 16
5.1.3.7 CL_SPECIALIZE_COL_TRAITS() [7/11] . . . 16
5.1.3.8 CL_SPECIALIZE_COL_TRAITS() [8/11] . . . 16
5.1.3.9 CL_SPECIALIZE_COL_TRAITS() [9/11] . . . 16
5.1.3.10 CL_SPECIALIZE_COL_TRAITS() [10/11] . . . 16
5.1.3.11 CL_SPECIALIZE_COL_TRAITS() [11/11] . . . 16
5.1.3.12 dataSetAccessor() . . . 16
5.1.3.13 dos2unix() . . . 17
5.1.3.14 isAccelerometer() . . . 18
5.1.3.15 isGyroscope() . . . 18
5.1.3.16 operator<<() [1/4] . . . 19
5.1.3.17 operator<<() [2/4] . . . 19
5.1.3.18 operator<<() [3/4] . . . 20
5.1.3.19 operator<<() [4/4] . . . 21
5.1.3.20 readCsvFile() . . . 21
5.1.3.21 s2n() . . . 22

5.1.3.22 threshold() . . . 22
5.1.3.23 to_string() . . . 23
5.1.3.24 useUnbufferedIo() . . . 24
5.1.4 Variable Documentation . . . 24
5.1.4.1 accelerometerThreshold . . . 24
5.1.4.2 channelCount . . . 24
5.1.4.3 channels . . . 25
5.1.4.4 column_index . . . 25
5.1.4.5 data_set_accessor_v . . . 25
5.1.4.6 gyroscopeThreshold . . . 25
5.1.4.7 sensors . . . 26
5.2 cl::fs Namespace Reference . . . 26
5.2.1 Enumeration Type Documentation . . . 26
5.2.1.1 DirectoryListingOption . . . 26
5.2.2 Function Documentation . . . 27
5.2.2.1 directoryListing() . . . 27
5.2.2.2 formatError() . . . 28
5.2.2.3 operator<() . . . 28
5.2.2.4 operator<<() . . . 29
5.2.2.5 operator==() . . . 29
5.2.2.6 utf16ToUtf8() . . . 30
5.2.2.7 utf8ToUtf16() . . . 30
5.3 cm Namespace Reference . . . 31
5.3.1 Enumeration Type Documentation . . . 33
5.3.1.1 DataSetIdentifier . . . 33
5.3.1.2 Imu . . . 33
5.3.2 Function Documentation . . . 34
5.3.2.1 closestOne() . . . 34
5.3.2.2 CM_SORTER() [1/4] . . . 35
5.3.2.3 CM_SORTER() [2/4] . . . 35
5.3.2.4 CM_SORTER() [3/4] . . . 35
5.3.2.5 CM_SORTER() [4/4] . . . 35
5.3.2.6 confusionMatrixBestConfigs() . . . 36
5.3.2.7 createSegmentationResults() . . . 37
5.3.2.8 distance() . . . 38
5.3.2.9 distanceScore() . . . 39
5.3.2.10 fetch() . . . 40
5.3.2.11 interpolatedDataSetPaths() . . . 41
5.3.2.12 operator"!=() . . . 42
5.3.2.13 operator<() [1/2] . . . 42
5.3.2.14 operator<() [2/2] . . . 43
5.3.2.15 operator<<() [1/6] . . . 43

5.3.2.16 operator<<() [2/6] . . . 43
5.3.2.17 operator<<() [3/6] . . . 43
5.3.2.18 operator<<() [4/6] . . . 44
5.3.2.19 operator<<() [5/6] . . . 45
5.3.2.20 operator<<() [6/6] . . . 45
5.3.2.21 operator==() [1/2] . . . 46
5.3.2.22 operator==() [2/2] . . . 46
5.3.2.23 orderConfigurationsByQuality() . . . 47
5.3.2.24 pythonOutput() . . . 47
5.3.2.25 segment() . . . 48
5.3.2.26 splitString() . . . 50
5.3.2.27 toDataSetIdentifier() . . . 50
5.3.3 Variable Documentation . . . 51
5.3.3.1 addTrueSubtractFalseSorter . . . 51
5.3.3.2 disregardTrueNegativesSorter . . . 51
5.3.3.3 imuCount . . . 52
5.3.3.4 imus . . . 52
5.4 cs Namespace Reference . . . 52
5.4.1 Enumeration Type Documentation . . . 53
5.4.1.1 FilterKind . . . 53
5.4.1.2 Mode . . . 54
5.4.1.3 SegmentationKind . . . 54
5.4.2 Function Documentation . . . 54
5.4.2.1 CS_SPECIALIZE_DATA_SET_INFO() [1/20] . . . 54
5.4.2.2 CS_SPECIALIZE_DATA_SET_INFO() [2/20] . . . 55
5.4.2.3 CS_SPECIALIZE_DATA_SET_INFO() [3/20] . . . 55
5.4.2.4 CS_SPECIALIZE_DATA_SET_INFO() [4/20] . . . 55
5.4.2.5 CS_SPECIALIZE_DATA_SET_INFO() [5/20] . . . 55
5.4.2.6 CS_SPECIALIZE_DATA_SET_INFO() [6/20] . . . 55
5.4.2.7 CS_SPECIALIZE_DATA_SET_INFO() [7/20] . . . 56
5.4.2.8 CS_SPECIALIZE_DATA_SET_INFO() [8/20] . . . 56
5.4.2.9 CS_SPECIALIZE_DATA_SET_INFO() [9/20] . . . 56
5.4.2.10 CS_SPECIALIZE_DATA_SET_INFO() [10/20] . . . 56
5.4.2.11 CS_SPECIALIZE_DATA_SET_INFO() [11/20] . . . 56
5.4.2.12 CS_SPECIALIZE_DATA_SET_INFO() [12/20] . . . 56
5.4.2.13 CS_SPECIALIZE_DATA_SET_INFO() [13/20] . . . 57
5.4.2.14 CS_SPECIALIZE_DATA_SET_INFO() [14/20] . . . 57
5.4.2.15 CS_SPECIALIZE_DATA_SET_INFO() [15/20] . . . 57
5.4.2.16 CS_SPECIALIZE_DATA_SET_INFO() [16/20] . . . 57
5.4.2.17 CS_SPECIALIZE_DATA_SET_INFO() [17/20] . . . 57
5.4.2.18 CS_SPECIALIZE_DATA_SET_INFO() [18/20] . . . 57
5.4.2.19 CS_SPECIALIZE_DATA_SET_INFO() [19/20] . . . 58

5.4.2.20 CS_SPECIALIZE_DATA_SET_INFO() [20/20] . . . 58
5.4.2.21 logFiles() . . . 58
5.4.2.22 operator"!=() . . . 59
5.4.2.23 operator<<() [1/4] . . . 59
5.4.2.24 operator<<() [2/4] . . . 60
5.4.2.25 operator<<() [3/4] . . . 60
5.4.2.26 operator<<() [4/4] . . . 61
5.4.2.27 operator==() . . . 61
5.4.2.28 parseMode() . . . 61
5.4.2.29 PL_DEFINE_EXCEPTION_TYPE() . . . 62
5.4.2.30 repetitionCount() . . . 62
5.4.3 Variable Documentation . . . 63
5.4.3.1 logPath . . . 63
5.4.3.2 oldLogPath . . . 63
5.5 ctg Namespace Reference . . . 63
5.5.1 Function Documentation . . . 64
5.5.1.1 aboveThreshold() . . . 64
5.5.1.2 averageComparisonValueCalculator() . . . 65
5.5.1.3 halfMaximumComparisonValueCalculator() . . . 65
5.5.1.4 isRelevant() . . . 66
5.5.1.5 percentageOf() . . . 67
5.5.1.6 runAboveThreshold() . . . 68
5.6 fmc Namespace Reference . . . 68
5.6.1 Function Documentation . . . 69
5.6.1.1 adjustHardwareTimestamp() . . . 69
5.6.1.2 convertToUnixLineEndings() . . . 69
5.6.1.3 createBackupFile() . . . 70
5.6.1.4 deleteNonBoschSensors() . . . 71
5.6.1.5 deleteOutOfBoundsValues() . . . 71
5.6.1.6 removeZerosFromField() . . . 71
5.6.1.7 restoreFromBackup() . . . 72
5.6.1.8 writeFile() . . . 72

6 Class Documentation **75**
6.1 cm::Configuration::Builder Class Reference . . . 75
6.1.1 Detailed Description . . . 75
6.1.2 Constructor & Destructor Documentation . . . 76
6.1.2.1 Builder() . . . 76
6.1.3 Member Function Documentation . . . 76
6.1.3.1 build() . . . 76
6.1.3.2 deleteTooClose() . . . 77
6.1.3.3 deleteTooLowVariance() . . . 78

6.1.3.4 filterKind() . . . 79
6.1.3.5 imu() . . . 80
6.1.3.6 segmentationKind() . . . 81
6.1.3.7 skipWindow() . . . 82
6.1.3.8 windowSize() . . . 83
6.2 cl::col_traits< Col > Struct Template Reference . . . 84
6.2.1 Detailed Description . . . 84
6.3 cm::Configuration Class Reference . . . 85
6.3.1 Detailed Description . . . 86
6.3.2 Constructor & Destructor Documentation . . . 86
6.3.2.1 Configuration() . . . 86
6.3.3 Member Function Documentation . . . 87
6.3.3.1 createFilePath() . . . 87
6.3.3.2 deleteTooClose() . . . 87
6.3.3.3 deleteTooCloseOptions() . . . 88
6.3.3.4 deleteTooLowVariance() . . . 88
6.3.3.5 deleteTooLowVarianceOptions() . . . 89
6.3.3.6 filterKind() . . . 89
6.3.3.7 filterKindOptions() . . . 90
6.3.3.8 importSegmentationPoints() . . . 90
6.3.3.9 imu() . . . 91
6.3.3.10 imuOptions() . . . 92
6.3.3.11 isInitialized() . . . 92
6.3.3.12 segmentationKind() . . . 93
6.3.3.13 segmentationKindOptions() . . . 93
6.3.3.14 serializeSegmentationPoints() . . . 93
6.3.3.15 skipWindow() . . . 94
6.3.3.16 skipWindowOptions() . . . 95
6.3.3.17 windowSize() . . . 95
6.3.3.18 windowSizeOptions() . . . 96
6.3.4 Friends And Related Function Documentation . . . 96
6.3.4.1 Builder . . . 96
6.3.4.2 operator< . . . 96
6.3.4.3 operator<< . . . 97
6.3.4.4 operator== . . . 97
6.3.4.5 std::hash< Configuration > . . . 98
6.4 cm::ConfigWithDistanceScore Struct Reference . . . 98
6.4.1 Detailed Description . . . 98
6.4.2 Constructor & Destructor Documentation . . . 98
6.4.2.1 ConfigWithDistanceScore() . . . 99
6.4.3 Member Data Documentation . . . 99
6.4.3.1 config . . . 99

6.4.3.2 distScore . . . 99
6.5 cm::ConfigWithTotalConfusionMatrix Struct Reference . . . 99
6.5.1 Detailed Description . . . 100
6.5.2 Constructor & Destructor Documentation . . . 100
6.5.2.1 ConfigWithTotalConfusionMatrix() [1/2] . . . 100
6.5.2.2 ConfigWithTotalConfusionMatrix() [2/2] . . . 100
6.5.3 Friends And Related Function Documentation . . . 101
6.5.3.1 operator<< . . . 101
6.5.4 Member Data Documentation . . . 101
6.5.4.1 config . . . 101
6.5.4.2 matrix . . . 101
6.6 cm::ConfusionMatrix Class Reference . . . 102
6.6.1 Detailed Description . . . 102
6.6.2 Member Typedef Documentation . . . 102
6.6.2.1 this_type . . . 103
6.6.3 Constructor & Destructor Documentation . . . 103
6.6.3.1 ConfusionMatrix() . . . 103
6.6.4 Member Function Documentation . . . 103
6.6.4.1 falseNegatives() . . . 103
6.6.4.2 falsePositives() . . . 103
6.6.4.3 incrementFalseNegatives() . . . 104
6.6.4.4 incrementFalsePositives() . . . 104
6.6.4.5 incrementTrueNegatives() . . . 105
6.6.4.6 incrementTruePositives() . . . 105
6.6.4.7 operator+=() . . . 105
6.6.4.8 totalCount() . . . 106
6.6.4.9 trueNegatives() . . . 106
6.6.4.10 truePositives() . . . 107
6.7 cm::CsvFileInfo Class Reference . . . 107
6.7.1 Detailed Description . . . 107
6.7.2 Constructor & Destructor Documentation . . . 107
6.7.2.1 CsvFileInfo() . . . 107
6.7.3 Member Function Documentation . . . 108
6.7.3.1 hardwareTimestamps() . . . 108
6.8 cs::CsvLineBuilder Class Reference . . . 108
6.8.1 Detailed Description . . . 109
6.8.2 Member Typedef Documentation . . . 109
6.8.2.1 this_type . . . 109
6.8.3 Constructor & Destructor Documentation . . . 110
6.8.3.1 CsvLineBuilder() . . . 110
6.8.4 Member Function Documentation . . . 110
6.8.4.1 build() . . . 110

6.8.4.2 dataSet() . . . 111
6.8.4.3 deleteLowVariance() . . . 111
6.8.4.4 deleteTooClose() . . . 112
6.8.4.5 filter() . . . 113
6.8.4.6 isOld() . . . 114
6.8.4.7 kind() . . . 115
6.8.4.8 repetitions() . . . 116
6.8.4.9 segmentationPoints() . . . 117
6.8.4.10 sensor() . . . 118
6.8.4.11 skipWindow() . . . 119
6.8.4.12 windowSize() . . . 120
6.9 cl::data_set_accessor< Chan > Struct Template Reference . . . 121
6.9.1 Detailed Description . . . 121
6.10 cs::data_set_info< Tag > Struct Template Reference . . . 122
6.10.1 Detailed Description . . . 122
6.11 cl::DataPoint Class Reference . . . 122
6.11.1 Detailed Description . . . 123
6.11.2 Constructor & Destructor Documentation . . . 123
6.11.2.1 DataPoint() . . . 123
6.11.3 Member Function Documentation . . . 124
6.11.3.1 channel() . . . 124
6.11.3.2 fileName() . . . 125
6.11.3.3 sensor() . . . 125
6.11.3.4 time() . . . 126
6.11.3.5 value() . . . 126
6.11.4 Friends And Related Function Documentation . . . 127
6.11.4.1 operator<< . . . 127
6.12 cl::DataSet Class Reference . . . 127
6.12.1 Detailed Description . . . 128
6.12.2 Member Typedef Documentation . . . 128
6.12.2.1 ChannelAccessor . . . 128
6.12.2.2 size_type . . . 128
6.12.3 Member Function Documentation . . . 128
6.12.3.1 accelerometerAverage() . . . 129
6.12.3.2 accelerometerMaximum() . . . 129
6.12.3.3 accelerometerX() . . . 130
6.12.3.4 accelerometerY() . . . 130
6.12.3.5 accelerometerZ() . . . 131
6.12.3.6 create() . . . 131
6.12.3.7 extractId() . . . 133
6.12.3.8 fileName() . . . 133
6.12.3.9 gyroscopeAverage() . . . 134

6.12.3.10 gyroscopeMaximum() . . . 134
6.12.3.11 gyroscopeX() . . . 135
6.12.3.12 gyroscopeY() . . . 135
6.12.3.13 gyroscopeZ() . . . 136
6.12.3.14 hardwareTimestamp() . . . 136
6.12.3.15 rowCount() . . . 137
6.12.3.16 time() . . . 137
6.12.3.17 trigger() . . . 137
6.13 cl::Error Class Reference . . . 138
6.13.1 Detailed Description . . . 138
6.13.2 Member Enumeration Documentation . . . 138
6.13.2.1 Kind . . . 138
6.13.3 Constructor & Destructor Documentation . . . 139
6.13.3.1 Error() . . . 139
6.13.4 Member Function Documentation . . . 139
6.13.4.1 file() . . . 139
6.13.4.2 function() . . . 140
6.13.4.3 kind() . . . 140
6.13.4.4 line() . . . 140
6.13.4.5 message() . . . 141
6.13.4.6 raise() . . . 141
6.13.4.7 to_string() . . . 141
6.13.5 Friends And Related Function Documentation . . . 141
6.13.5.1 operator<< . . . 141
6.14 cl::Exception Class Reference . . . 142
6.14.1 Detailed Description . . . 143
6.14.2 Member Typedef Documentation . . . 143
6.14.2.1 base_type . . . 143
6.14.3 Constructor & Destructor Documentation . . . 143
6.14.3.1 Exception() [1/2] . . . 143
6.14.3.2 Exception() [2/2] . . . 143
6.14.4 Member Function Documentation . . . 143
6.14.4.1 file() . . . 144
6.14.4.2 function() . . . 144
6.14.4.3 line() . . . 144
6.15 cl::fs::File Class Reference . . . 145
6.15.1 Detailed Description . . . 145
6.15.2 Constructor & Destructor Documentation . . . 146
6.15.2.1 File() . . . 146
6.15.3 Member Function Documentation . . . 146
6.15.3.1 copyTo() . . . 146
6.15.3.2 create() . . . 147

6.15.3.3 exists() . . . 148
6.15.3.4 moveTo() . . . 148
6.15.3.5 path() . . . 149
6.15.3.6 remove() . . . 150
6.15.3.7 size() . . . 150
6.16 cl::fs::FileStream Class Reference . . . 151
6.16.1 Detailed Description . . . 152
6.16.2 Member Typedef Documentation . . . 152
6.16.2.1 this_type . . . 152
6.16.3 Member Enumeration Documentation . . . 152
6.16.3.1 OpenMode . . . 152
6.16.4 Constructor & Destructor Documentation . . . 153
6.16.4.1 FileStream() . . . 153
6.16.4.2 ∼FileStream() . . . 153
6.16.5 Member Function Documentation . . . 153
6.16.5.1 create() . . . 153
6.16.5.2 operator=() . . . 154
6.16.5.3 PL_NONCOPYABLE() . . . 155
6.16.5.4 readAll() . . . 155
6.16.5.5 write() . . . 155
6.17 std::hash<::cl::fs::Path > Struct Reference . . . 156
6.17.1 Detailed Description . . . 156
6.17.2 Member Function Documentation . . . 156
6.17.2.1 operator()() . . . 156
6.18 std::hash<::cm::Configuration > Struct Reference . . . 156
6.18.1 Detailed Description . . . 156
6.18.2 Member Function Documentation . . . 156
6.18.2.1 operator()() . . . 157
6.19 cs::LogInfo Class Reference . . . 157
6.19.1 Detailed Description . . . 158
6.19.2 Constructor & Destructor Documentation . . . 158
6.19.2.1 LogInfo() . . . 158
6.19.3 Member Function Documentation . . . 158
6.19.3.1 create() . . . 158
6.19.3.2 deleteLowVariance() . . . 159
6.19.3.3 deleteTooClose() . . . 159
6.19.3.4 filterKind() . . . 160
6.19.3.5 isInitialized() . . . 160
6.19.3.6 logFilePath() . . . 160
6.19.3.7 segmentationKind() . . . 161
6.19.3.8 sensor() . . . 161
6.19.3.9 skipWindow() . . . 161

6.19.3.10 windowSize() . . . 162
6.19.4 Friends And Related Function Documentation . . . 162
6.19.4.1 operator"!= . . . 162
6.19.4.2 operator<< . . . 162
6.19.4.3 operator== . . . 163
6.19.5 Member Data Documentation . . . 163
6.19.5.1 invalidSensor . . . 163
6.20 cs::LogLine Class Reference . . . 163
6.20.1 Detailed Description . . . 164
6.20.2 Member Function Documentation . . . 164
6.20.2.1 fileName() . . . 164
6.20.2.2 filePath() . . . 165
6.20.2.3 parse() . . . 166
6.20.2.4 segmentationPointCount() . . . 166
6.20.2.5 sensor() . . . 167
6.20.3 Member Data Documentation . . . 167
6.20.3.1 invalidSensor . . . 167
6.21 cm::ManualSegmentationPoint Class Reference . . . 167
6.21.1 Detailed Description . . . 168
6.21.2 Constructor & Destructor Documentation . . . 168
6.21.2.1 ManualSegmentationPoint() . . . 168
6.21.3 Member Function Documentation . . . 169
6.21.3.1 asMilliseconds() . . . 169
6.21.3.2 convertToHardwareTimestamps() . . . 169
6.21.3.3 frame() . . . 170
6.21.3.4 hour() . . . 171
6.21.3.5 minute() . . . 172
6.21.3.6 readCsvFile() . . . 172
6.21.3.7 second() . . . 173
6.21.4 Friends And Related Function Documentation . . . 173
6.21.4.1 operator"!= . . . 173
6.21.4.2 operator<< . . . 174
6.21.4.3 operator== . . . 174
6.22 cl::fs::Path Class Reference . . . 175
6.22.1 Detailed Description . . . 175
6.22.2 Constructor & Destructor Documentation . . . 176
6.22.2.1 Path() [1/3] . . . 176
6.22.2.2 Path() [2/3] . . . 176
6.22.2.3 Path() [3/3] . . . 176
6.22.3 Member Function Documentation . . . 176
6.22.3.1 exists() . . . 177
6.22.3.2 isDirectory() . . . 177

6.22.3.3 isFile() . . . 178
6.22.3.4 str() . . . 179
6.22.4 Friends And Related Function Documentation . . . 179
6.22.4.1 operator< . . . 179
6.22.4.2 operator<< . . . 180
6.22.4.3 operator== . . . 180
6.23 cl::Process Class Reference . . . 181
6.23.1 Detailed Description . . . 181
6.23.2 Member Typedef Documentation . . . 181
6.23.2.1 this_type . . . 181
6.23.3 Constructor & Destructor Documentation . . . 181
6.23.3.1 Process() . . . 181
6.23.3.2 ~Process() . . . 182
6.23.4 Member Function Documentation . . . 182
6.23.4.1 create() . . . 182
6.23.4.2 file() [1/2] . . . 182
6.23.4.3 file() [2/2] . . . 182
6.23.4.4 operator=() . . . 182
6.23.4.5 PL_NONCOPYABLE() . . . 183

7 File Documentation **185**
7.1 compare_segmentation/CMakeLists.txt File Reference . . . 185
7.1.1 Function Documentation . . . 185
7.1.1.1 set() . . . 185
7.2 compare_segmentation/test/CMakeLists.txt File Reference . . . 185
7.2.1 Function Documentation . . . 185
7.2.1.1 include() . . . 186
7.3 counting/CMakeLists.txt File Reference . . . 186
7.3.1 Function Documentation . . . 186
7.3.1.1 set() . . . 186
7.4 counting/test/CMakeLists.txt File Reference . . . 186
7.4.1 Function Documentation . . . 186
7.4.1.1 include() . . . 186
7.5 csv_lib/CMakeLists.txt File Reference . . . 187
7.5.1 Function Documentation . . . 187
7.5.1.1 set() . . . 187
7.6 csv_lib/test/CMakeLists.txt File Reference . . . 187
7.6.1 Function Documentation . . . 187
7.6.1.1 include() . . . 187
7.7 fix_csv/CMakeLists.txt File Reference . . . 188
7.7.1 Function Documentation . . . 188
7.7.1.1 set() . . . 188

7.8 fix_csv/test/CMakeLists.txt File Reference . . . 188
7.8.1 Function Documentation . . . 188
7.8.1.1 include() . . . 188
7.9 confusion_matrix/CMakeLists.txt File Reference . . . 189
7.9.1 Function Documentation . . . 189
7.9.1.1 set() . . . 189
7.10 confusion_matrix/test/CMakeLists.txt File Reference . . . 189
7.10.1 Function Documentation . . . 189
7.10.1.1 include() . . . 189
7.11 compare_segmentation/include/csv_line.hpp File Reference . . . 190
7.12 compare_segmentation/include/data_set_info.hpp File Reference . . . 191
7.12.1 Macro Definition Documentation . . . 192
7.12.1.1 CS_SPECIALIZE_DATA_SET_INFO . . . 192
7.13 compare_segmentation/include/filter_kind.hpp File Reference . . . 193
7.14 compare_segmentation/include/log_files.hpp File Reference . . . 194
7.15 compare_segmentation/include/log_info.hpp File Reference . . . 194
7.16 compare_segmentation/include/log_line.hpp File Reference . . . 195
7.17 compare_segmentation/include/mode.hpp File Reference . . . 196
7.17.1 Macro Definition Documentation . . . 197
7.17.1.1 CS_MODE . . . 197
7.17.1.2 CS_MODE_X . . . 197
7.18 compare_segmentation/include/paths.hpp File Reference . . . 198
7.19 compare_segmentation/include/segmentation_kind.hpp File Reference . . . 199
7.20 compare_segmentation/src/csv_line.cpp File Reference . . . 200
7.21 compare_segmentation/src/data_set_info.cpp File Reference . . . 200
7.22 compare_segmentation/src/filter_kind.cpp File Reference . . . 201
7.23 compare_segmentation/src/log_files.cpp File Reference . . . 201
7.24 compare_segmentation/src/log_info.cpp File Reference . . . 202
7.25 compare_segmentation/src/log_line.cpp File Reference . . . 203
7.26 compare_segmentation/src/main.cpp File Reference . . . 203
7.26.1 Function Documentation . . . 204
7.26.1.1 main() . . . 204
7.27 compare_segmentation/test/main.cpp File Reference . . . 205
7.27.1 Function Documentation . . . 206
7.27.1.1 main() . . . 206
7.28 counting/src/main.cpp File Reference . . . 206
7.28.1 Function Documentation . . . 207
7.28.1.1 main() . . . 207
7.29 counting/test/main.cpp File Reference . . . 208
7.29.1 Function Documentation . . . 209
7.29.1.1 main() . . . 209
7.30 csv_lib/test/main.cpp File Reference . . . 209

7.30.1 Function Documentation . . . 210
7.30.1.1 main() . . . 210
7.31 fix_csv/src/main.cpp File Reference . . . 210
7.31.1 Function Documentation . . . 211
7.31.1.1 main() . . . 211
7.32 fix_csv/test/main.cpp File Reference . . . 211
7.32.1 Function Documentation . . . 212
7.32.1.1 main() . . . 212
7.33 confusion_matrix/src/main.cpp File Reference . . . 212
7.33.1 Macro Definition Documentation . . . 213
7.33.1.1 SORT_PRINT . . . 213
7.33.2 Function Documentation . . . 213
7.33.2.1 main() . . . 213
7.34 confusion_matrix/test/main.cpp File Reference . . . 214
7.34.1 Function Documentation . . . 215
7.34.1.1 main() . . . 215
7.35 compare_segmentation/src/mode.cpp File Reference . . . 215
7.35.1 Macro Definition Documentation . . . 216
7.35.1.1 CS_MODE_X [1/2] . . . 216
7.35.1.2 CS_MODE_X [2/2] . . . 216
7.36 compare_segmentation/src/segmentation_kind.cpp File Reference . . . 217
7.37 compare_segmentation/test/csv_line_test.cpp File Reference . . . 217
7.37.1 Function Documentation . . . 218
7.37.1.1 TEST() . . . 218
7.38 compare_segmentation/test/data_set_info_test.cpp File Reference . . . 218
7.38.1 Function Documentation . . . 218
7.38.1.1 TEST() . . . 219
7.39 compare_segmentation/test/log_files_test.cpp File Reference . . . 219
7.39.1 Function Documentation . . . 219
7.39.1.1 TEST() [1/3] . . . 220
7.39.1.2 TEST() [2/3] . . . 220
7.39.1.3 TEST() [3/3] . . . 221
7.40 compare_segmentation/test/log_info_test.cpp File Reference . . . 221
7.40.1 Function Documentation . . . 222
7.40.1.1 TEST() [1/19] . . . 222
7.40.1.2 TEST() [2/19] . . . 222
7.40.1.3 TEST() [3/19] . . . 223
7.40.1.4 TEST() [4/19] . . . 223
7.40.1.5 TEST() [5/19] . . . 224
7.40.1.6 TEST() [6/19] . . . 224
7.40.1.7 TEST() [7/19] . . . 225
7.40.1.8 TEST() [8/19] . . . 225

7.40.1.9 TEST() [9/19] . . . 226
7.40.1.10 TEST() [10/19] . . . 226
7.40.1.11 TEST() [11/19] . . . 227
7.40.1.12 TEST() [12/19] . . . 227
7.40.1.13 TEST() [13/19] . . . 228
7.40.1.14 TEST() [14/19] . . . 228
7.40.1.15 TEST() [15/19] . . . 229
7.40.1.16 TEST() [16/19] . . . 229
7.40.1.17 TEST() [17/19] . . . 230
7.40.1.18 TEST() [18/19] . . . 230
7.40.1.19 TEST() [19/19] . . . 231
7.41 compare_segmentation/test/log_line_test.cpp File Reference . . . 231
7.41.1 Function Documentation . . . 231
7.41.1.1 TEST() [1/4] . . . 232
7.41.1.2 TEST() [2/4] . . . 232
7.41.1.3 TEST() [3/4] . . . 233
7.41.1.4 TEST() [4/4] . . . 233
7.42 compare_segmentation/test/mode_test.cpp File Reference . . . 233
7.42.1 Function Documentation . . . 234
7.42.1.1 TEST() [1/2] . . . 234
7.42.1.2 TEST() [2/2] . . . 235
7.43 confusion_matrix/include/closest_one.hpp File Reference . . . 235
7.44 confusion_matrix/include/configuration.hpp File Reference . . . 236
7.45 confusion_matrix/include/confusion_matrix.hpp File Reference . . . 237
7.46 confusion_matrix/include/confusion_matrix_best_configs.hpp File Reference . . . 238
7.46.1 Macro Definition Documentation . . . 240
7.46.1.1 CM_SORTER . . . 240
7.47 confusion_matrix/include/create_segmentation_results.hpp File Reference . . . 240
7.48 confusion_matrix/include/csv_file_info.hpp File Reference . . . 241
7.49 confusion_matrix/include/data_set_identifier.hpp File Reference . . . 242
7.49.1 Macro Definition Documentation . . . 244
7.49.1.1 CM_DATA_SET_IDENTIFIER . . . 244
7.49.1.2 CM_DATA_SET_IDENTIFIER_X . . . 244
7.50 confusion_matrix/include/distance.hpp File Reference . . . 244
7.51 confusion_matrix/include/distance_score.hpp File Reference . . . 245
7.52 confusion_matrix/include/fetch.hpp File Reference . . . 246
7.53 confusion_matrix/include/imu.hpp File Reference . . . 247
7.53.1 Macro Definition Documentation . . . 248
7.53.1.1 CM_IMU . . . 249
7.53.1.2 CM_IMU_X [1/3] . . . 249
7.53.1.3 CM_IMU_X [2/3] . . . 249
7.53.1.4 CM_IMU_X [3/3] . . . 249

7.54 confusion_matrix/include/interpolated_data_set_paths.hpp File Reference . . . 250
7.55 confusion_matrix/include/manual_segmentation_point.hpp File Reference . . . 251
7.56 confusion_matrix/include/order_configurations_by_quality.hpp File Reference . . . 252
7.57 confusion_matrix/include/python_output.hpp File Reference . . . 253
7.58 confusion_matrix/include/segment.hpp File Reference . . . 254
7.59 confusion_matrix/include/split_string.hpp File Reference . . . 255
7.60 confusion_matrix/src/closest_one.cpp File Reference . . . 255
7.61 confusion_matrix/src/configuration.cpp File Reference . . . 256
7.61.1 Macro Definition Documentation . . . 257
7.61.1.1 CM_ENSURE_CONTAINS . . . 257
7.61.1.2 CM_ENSURE_HAS_VALUE . . . 257
7.62 confusion_matrix/src/confusion_matrix.cpp File Reference . . . 258
7.63 confusion_matrix/src/confusion_matrix_best_configs.cpp File Reference . . . 258
7.64 confusion_matrix/src/create_segmentation_results.cpp File Reference . . . 259
7.65 confusion_matrix/src/csv_file_info.cpp File Reference . . . 260
7.66 confusion_matrix/src/data_set_identifier.cpp File Reference . . . 260
7.66.1 Macro Definition Documentation . . . 261
7.66.1.1 CM_DATA_SET_IDENTIFIER_X . . . 261
7.66.1.2 DSI . . . 261
7.67 confusion_matrix/src/distance.cpp File Reference . . . 262
7.68 confusion_matrix/src/distance_score.cpp File Reference . . . 262
7.69 confusion_matrix/src/imu.cpp File Reference . . . 263
7.69.1 Macro Definition Documentation . . . 263
7.69.1.1 CM_IMU_X . . . 264
7.70 confusion_matrix/src/interpolated_data_set_paths.cpp File Reference . . . 264
7.71 confusion_matrix/src/manual_segmentation_point.cpp File Reference . . . 264
7.71.1 Macro Definition Documentation . . . 265
7.71.1.1 DSI . . . 265
7.72 confusion_matrix/src/order_configurations_by_quality.cpp File Reference . . . 266
7.73 confusion_matrix/src/python_output.cpp File Reference . . . 266
7.73.1 Macro Definition Documentation . . . 267
7.73.1.1 CM_DEV_NULL . . . 267
7.73.1.2 CM_SEGMENTOR . . . 267
7.74 confusion_matrix/src/segment.cpp File Reference . . . 267
7.75 confusion_matrix/src/split_string.cpp File Reference . . . 268
7.76 confusion_matrix/test/data_set_identifier_test.cpp File Reference . . . 269
7.76.1 Macro Definition Documentation . . . 269
7.76.1.1 DSI . . . 269
7.76.2 Function Documentation . . . 269
7.76.2.1 TEST() . . . 270
7.77 confusion_matrix/test/interpolated_data_set_paths_test.cpp File Reference . . . 270
7.77.1 Function Documentation . . . 270

7.77.1.1 TEST() . . . 271
7.78 confusion_matrix/test/manual_segmentation_point_test.cpp File Reference . . . 271
7.78.1 Macro Definition Documentation . . . 272
7.78.1.1 DSI . . . 272
7.78.2 Function Documentation . . . 272
7.78.2.1 TEST() [1/11] . . . 272
7.78.2.2 TEST() [2/11] . . . 273
7.78.2.3 TEST() [3/11] . . . 273
7.78.2.4 TEST() [4/11] . . . 273
7.78.2.5 TEST() [5/11] . . . 273
7.78.2.6 TEST() [6/11] . . . 273
7.78.2.7 TEST() [7/11] . . . 274
7.78.2.8 TEST() [8/11] . . . 274
7.78.2.9 TEST() [9/11] . . . 274
7.78.2.10 TEST() [10/11] . . . 274
7.78.2.11 TEST() [11/11] . . . 274
7.79 confusion_matrix/test/segment_test.cpp File Reference . . . 275
7.79.1 Macro Definition Documentation . . . 275
7.79.1.1 EXPECT_SEGMENTATION_POINTS . . . 275
7.79.2 Function Documentation . . . 275
7.79.2.1 TEST() . . . 276
7.80 confusion_matrix/test/split_string_test.cpp File Reference . . . 276
7.80.1 Function Documentation . . . 277
7.80.1.1 TEST() . . . 277
7.81 counting/include/above_threshold.hpp File Reference . . . 278
7.82 counting/include/average_comparison_value_calculator.hpp File Reference . . . 279
7.83 counting/include/half_maximum_comparison_value_calculator.hpp File Reference . . . 279
7.84 counting/include/is_relevant.hpp File Reference . . . 280
7.85 counting/include/percentage_of.hpp File Reference . . . 281
7.86 counting/include/run_above_threshold.hpp File Reference . . . 282
7.87 counting/src/above_threshold.cpp File Reference . . . 283
7.87.1 Macro Definition Documentation . . . 284
7.87.1.1 CL_CHANNEL_X . . . 284
7.87.2 Variable Documentation . . . 284
7.87.2.1 channel . . . 285
7.87.2.2 channelAccessor . . . 285
7.88 counting/src/average_comparison_value_calculator.cpp File Reference . . . 285
7.89 counting/src/half_maximum_comparison_value_calculator.cpp File Reference . . . 286
7.90 counting/src/run_above_threshold.cpp File Reference . . . 286
7.91 counting/test/above_threshold_test.cpp File Reference . . . 287
7.91.1 Macro Definition Documentation . . . 287
7.91.1.1 EXPECT_LONG_DOUBLE_EQ . . . 288

7.91.2 Function Documentation . . . 288
7.91.2.1 TEST() . . . 288
7.92 counting/test/percentage_of_test.cpp File Reference . . . 289
7.92.1 Macro Definition Documentation . . . 289
7.92.1.1 EXPECT_LONG_DOUBLE_EQ . . . 289
7.92.2 Function Documentation . . . 289
7.92.2.1 TEST() . . . 290
7.93 csv_lib/include/cl/channel.hpp File Reference . . . 290
7.93.1 Macro Definition Documentation . . . 292
7.93.1.1 CL_CHANNEL . . . 292
7.93.1.2 CL_CHANNEL_X [1/4] . . . 292
7.93.1.3 CL_CHANNEL_X [2/4] . . . 292
7.93.1.4 CL_CHANNEL_X [3/4] . . . 292
7.93.1.5 CL_CHANNEL_X [4/4] . . . 293
7.94 csv_lib/include/cl/column.hpp File Reference . . . 293
7.94.1 Detailed Description . . . 295
7.94.2 Macro Definition Documentation . . . 295
7.94.2.1 CL_SPECIALIZE_COL_TRAITS . . . 295
7.95 csv_lib/include/cl/data_point.hpp File Reference . . . 295
7.96 csv_lib/include/cl/data_set.hpp File Reference . . . 296
7.97 csv_lib/include/cl/dos2unix.hpp File Reference . . . 297
7.98 csv_lib/include/cl/error.hpp File Reference . . . 298
7.98.1 Macro Definition Documentation . . . 298
7.98.1.1 CL_ERROR_KIND . . . 299
7.98.1.2 CL_ERROR_KIND_X . . . 299
7.98.1.3 CL_UNEXPECTED . . . 299
7.99 csv_lib/include/cl/exception.hpp File Reference . . . 299
7.99.1 Macro Definition Documentation . . . 300
7.99.1.1 CL_THROW . . . 300
7.99.1.2 CL_THROW_FMT . . . 300
7.100 csv_lib/include/cl/fs/directory_listing.hpp File Reference . . . 301
7.101 csv_lib/include/cl/fs/file.hpp File Reference . . . 302
7.102 csv_lib/include/cl/fs/file_stream.hpp File Reference . . . 302
7.103 csv_lib/include/cl/fs/path.hpp File Reference . . . 303
7.104 csv_lib/include/cl/fs/separator.hpp File Reference . . . 304
7.104.1 Macro Definition Documentation . . . 305
7.104.1.1 CL_FS_SEPARATOR . . . 305
7.105 csv_lib/include/cl/fs/windows.hpp File Reference . . . 305
7.105.1 Detailed Description . . . 306
7.106 csv_lib/include/cl/process.hpp File Reference . . . 307
7.107 csv_lib/include/cl/read_csv_file.hpp File Reference . . . 307
7.108 csv_lib/include/cl/s2n.hpp File Reference . . . 308

7.109 csv_lib/include/cl/sensor.hpp File Reference . . . 309
7.109.1 Macro Definition Documentation . . . 310
7.109.1.1 CL_SENSOR . . . 310
7.109.1.2 CL_SENSOR_X [1/2] . . . 311
7.109.1.3 CL_SENSOR_X [2/2] . . . 311
7.110 csv_lib/include/cl/to_string.hpp File Reference . . . 311
7.111 csv_lib/include/cl/use_unbuffered_io.hpp File Reference . . . 312
7.112 csv_lib/src/cl/channel.cpp File Reference . . . 312
7.112.1 Macro Definition Documentation . . . 313
7.112.1.1 CL_CHANNEL_X [1/2] . . . 313
7.112.1.2 CL_CHANNEL_X [2/2] . . . 313
7.113 csv_lib/src/cl/data_point.cpp File Reference . . . 314
7.113.1 Function Documentation . . . 314
7.113.1.1 channel() . . . 314
7.113.1.2 fileName() . . . 315
7.113.1.3 sensor() . . . 315
7.113.1.4 time() . . . 316
7.113.1.5 value() . . . 316
7.114 csv_lib/src/cl/data_set.cpp File Reference . . . 317
7.115 csv_lib/src/cl/dos2unix.cpp File Reference . . . 317
7.116 csv_lib/src/cl/error.cpp File Reference . . . 318
7.116.1 Macro Definition Documentation . . . 318
7.116.1.1 CL_ERROR_KIND_X . . . 319
7.117 csv_lib/src/cl/exception.cpp File Reference . . . 319
7.118 csv_lib/src/cl/fs/directory_listing.cpp File Reference . . . 319
7.119 csv_lib/src/cl/fs/file.cpp File Reference . . . 320
7.120 csv_lib/src/cl/fs/file_stream.cpp File Reference . . . 320
7.121 csv_lib/src/cl/fs/path.cpp File Reference . . . 321
7.122 csv_lib/src/cl/fs/windows.cpp File Reference . . . 322
7.123 csv_lib/src/cl/process.cpp File Reference . . . 322
7.124 csv_lib/src/cl/read_csv_file.cpp File Reference . . . 323
7.125 csv_lib/src/cl/sensor.cpp File Reference . . . 324
7.125.1 Macro Definition Documentation . . . 324
7.125.1.1 CL_SENSOR_X . . . 324
7.126 csv_lib/src/cl/use_unbuffered_io.cpp File Reference . . . 325
7.127 csv_lib/test/channel_test.cpp File Reference . . . 325
7.127.1 Function Documentation . . . 326
7.127.1.1 TEST() [1/4] . . . 326
7.127.1.2 TEST() [2/4] . . . 326
7.127.1.3 TEST() [3/4] . . . 326
7.127.1.4 TEST() [4/4] . . . 327
7.128 csv_lib/test/column_test.cpp File Reference . . . 327

7.128.1 Function Documentation . . . 328
7.128.1.1 TEST() [1/2] . . . 328
7.128.1.2 TEST() [2/2] . . . 328
7.129 csv_lib/test/data_point_test.cpp File Reference . . . 329
7.129.1 Function Documentation . . . 329
7.129.1.1 TEST() [1/2] . . . 329
7.129.1.2 TEST() [2/2] . . . 330
7.129.2 Variable Documentation . . . 330
7.129.2.1 dp . . . 330
7.130 csv_lib/test/data_set_test.cpp File Reference . . . 331
7.130.1 Macro Definition Documentation . . . 331
7.130.1.1 EXPECT_LONG_DOUBLE_EQ . . . 331
7.130.2 Function Documentation . . . 331
7.130.2.1 TEST() [1/4] . . . 332
7.130.2.2 TEST() [2/4] . . . 332
7.130.2.3 TEST() [3/4] . . . 333
7.130.2.4 TEST() [4/4] . . . 334
7.131 csv_lib/test/directory_listing_test.cpp File Reference . . . 335
7.131.1 Function Documentation . . . 336
7.131.1.1 TEST() [1/3] . . . 336
7.131.1.2 TEST() [2/3] . . . 336
7.131.1.3 TEST() [3/3] . . . 337
7.132 csv_lib/test/error_test.cpp File Reference . . . 337
7.132.1 Function Documentation . . . 338
7.132.1.1 TEST() [1/4] . . . 338
7.132.1.2 TEST() [2/4] . . . 338
7.132.1.3 TEST() [3/4] . . . 338
7.132.1.4 TEST() [4/4] . . . 338
7.132.2 Variable Documentation . . . 338
7.132.2.1 error . . . 339
7.133 csv_lib/test/exception_test.cpp File Reference . . . 339
7.133.1 Function Documentation . . . 339
7.133.1.1 TEST() . . . 339
7.134 csv_lib/test/read_csv_file_test.cpp File Reference . . . 340
7.134.1 Function Documentation . . . 340
7.134.1.1 TEST() [1/2] . . . 340
7.134.1.2 TEST() [2/2] . . . 341
7.135 csv_lib/test/s2n_test.cpp File Reference . . . 341
7.135.1 Function Documentation . . . 342
7.135.1.1 TEST() [1/3] . . . 342
7.135.1.2 TEST() [2/3] . . . 342
7.135.1.3 TEST() [3/3] . . . 343

7.136 csv_lib/test/sensor_test.cpp File Reference . . . 343
7.136.1 Function Documentation . . . 344
7.136.1.1 TEST() [1/2] . . . 344
7.136.1.2 TEST() [2/2] . . . 344
7.137 csv_lib/test/to_string_test.cpp File Reference . . . 344
7.137.1 Function Documentation . . . 345
7.137.1.1 TEST() . . . 345
7.138 fix_csv/include/adjust_hardware_timestamp.hpp File Reference . . . 345
7.139 fix_csv/include/convert_to_unix_line_endings.hpp File Reference . . . 346
7.140 fix_csv/include/create_backup_file.hpp File Reference . . . 347
7.141 fix_csv/include/delete_non_bosch_sensors.hpp File Reference . . . 348
7.142 fix_csv/include/delete_out_of_bounds_values.hpp File Reference . . . 349
7.143 fix_csv/include/remove_zeros_from_field.hpp File Reference . . . 350
7.144 fix_csv/include/restore_from_backup.hpp File Reference . . . 351
7.145 fix_csv/include/write_file.hpp File Reference . . . 352
7.146 fix_csv/src/adjust_hardware_timestamp.cpp File Reference . . . 353
7.147 fix_csv/src/convert_to_unix_line_endings.cpp File Reference . . . 354
7.148 fix_csv/src/create_backup_file.cpp File Reference . . . 355
7.149 fix_csv/src/delete_non_bosch_sensors.cpp File Reference . . . 355
7.149.1 Macro Definition Documentation . . . 356
7.149.1.1 CL_SENSOR_X . . . 356
7.150 fix_csv/src/delete_out_of_bounds_values.cpp File Reference . . . 356
7.151 fix_csv/src/remove_zeros_from_field.cpp File Reference . . . 357
7.152 fix_csv/src/restore_from_backup.cpp File Reference . . . 357
7.153 fix_csv/src/write_file.cpp File Reference . . . 358
7.154 fix_csv/test/adjust_hardware_timestamp_test.cpp File Reference . . . 359
7.154.1 Function Documentation . . . 359
7.154.1.1 TEST() [1/5] . . . 359
7.154.1.2 TEST() [2/5] . . . 360
7.154.1.3 TEST() [3/5] . . . 360
7.154.1.4 TEST() [4/5] . . . 361
7.154.1.5 TEST() [5/5] . . . 361
7.155 fix_csv/test/remove_zeros_from_field_test.cpp File Reference . . . 361
7.155.1 Function Documentation . . . 362
7.155.1.1 TEST() [1/6] . . . 362
7.155.1.2 TEST() [2/6] . . . 363
7.155.1.3 TEST() [3/6] . . . 363
7.155.1.4 TEST() [4/6] . . . 364
7.155.1.5 TEST() [5/6] . . . 364
7.155.1.6 TEST() [6/6] . . . 365

Index **367**

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

cl 11
cl::fs 26
cm 31
cs 52
ctg 63
fmc 68

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

cm::Configuration::Builder 75
cl::col_traits< Col > 84
cm::Configuration 85
cm::ConfigWithDistanceScore 98
cm::ConfigWithTotalConfusionMatrix 99
cm::ConfusionMatrix 102
cm::CsvFileInfo 107
cs::CsvLineBuilder 108
cl::data_set_accessor< Chan > 121
cs::data_set_info< Tag > 122
cl::DataPoint 122
cl::DataSet 127
cl::Error 138
std::exception
    std::runtime_error
        cl::Exception 142
cl::fs::File 145
cl::fs::FileStream 151
std::hash<::cl::fs::Path > 156
std::hash<::cm::Configuration > 156
cs::LogInfo 157
cs::LogLine 163
cm::ManualSegmentationPoint 167
cl::fs::Path 175
cl::Process 181

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

cm::Configuration::Builder
Builder type for `Configuration` 75
cl::col_traits< Col >
Column traits for the columns of the old, non-preprocessed CSV files. Includes the index (0 based) of the column and the C++ data type to be used for the cell contents in that column . . 84
cm::Configuration
Represents a possible configuration for the Python segmentor 85
cm::ConfigWithDistanceScore 98
cm::ConfigWithTotalConfusionMatrix
A `Configuration` with a `ConfusionMatrix` 99
cm::ConfusionMatrix
Type to represent a confusion matrix 102
cm::CsvFileInfo
Type to hold the hardware timestamps of a CSV file 107
cs::CsvLineBuilder
Builder for a CSV line 108
cl::data_set_accessor< Chan >
Maps a `Channel` to its associated accessor member function pointer of the `DataSet` type . 121
cs::data_set_info< Tag >
Meta function for data set tags 122
cl::DataPoint
Type to represent a data point in a data set 122
cl::DataSet 127
cl::Error 138
cl::Exception 142
cl::fs::File
Represents a file 145
cl::fs::FileStream
A binary file stream 151
std::hash<::cl::fs::Path > 156
std::hash<::cm::Configuration > 156
cs::LogInfo
Information about a log file 157
cs::LogLine
A line out of a log file 163

cm::ManualSegmentationPoint
Type used to represent a manual segmentation point 167
cl::fs::Path
A filesystem path 175
cl::Process 181

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

compare_segmentation/include/csv_line.hpp . . . 190
compare_segmentation/include/data_set_info.hpp . . . 191
compare_segmentation/include/filter_kind.hpp . . . 193
compare_segmentation/include/log_files.hpp . . . 194
compare_segmentation/include/log_info.hpp . . . 194
compare_segmentation/include/log_line.hpp . . . 195
compare_segmentation/include/mode.hpp . . . 196
compare_segmentation/include/paths.hpp . . . 198
compare_segmentation/include/segmentation_kind.hpp . . . 199
compare_segmentation/src/csv_line.cpp . . . 200
compare_segmentation/src/data_set_info.cpp . . . 200
compare_segmentation/src/filter_kind.cpp . . . 201
compare_segmentation/src/log_files.cpp . . . 201
compare_segmentation/src/log_info.cpp . . . 202
compare_segmentation/src/log_line.cpp . . . 203
compare_segmentation/src/main.cpp . . . 203
compare_segmentation/src/mode.cpp . . . 215
compare_segmentation/src/segmentation_kind.cpp . . . 217
compare_segmentation/test/csv_line_test.cpp . . . 217
compare_segmentation/test/data_set_info_test.cpp . . . 218
compare_segmentation/test/log_files_test.cpp . . . 219
compare_segmentation/test/log_info_test.cpp . . . 221
compare_segmentation/test/log_line_test.cpp . . . 231
compare_segmentation/test/main.cpp . . . 205
compare_segmentation/test/mode_test.cpp . . . 233
confusion_matrix/include/closest_one.hpp . . . 235
confusion_matrix/include/configuration.hpp . . . 236
confusion_matrix/include/confusion_matrix.hpp . . . 237
confusion_matrix/include/confusion_matrix_best_configs.hpp . . . 238
confusion_matrix/include/create_segmentation_results.hpp . . . 240
confusion_matrix/include/csv_file_info.hpp . . . 241
confusion_matrix/include/data_set_identifier.hpp . . . 242
confusion_matrix/include/distance.hpp . . . 244
confusion_matrix/include/distance_score.hpp . . . 245
confusion_matrix/include/fetch.hpp . . . 246

confusion_matrix/include/imu.hpp . . . 247
confusion_matrix/include/interpolated_data_set_paths.hpp . . . 250
confusion_matrix/include/manual_segmentation_point.hpp . . . 251
confusion_matrix/include/order_configurations_by_quality.hpp . . . 252
confusion_matrix/include/python_output.hpp . . . 253
confusion_matrix/include/segment.hpp . . . 254
confusion_matrix/include/split_string.hpp . . . 255
confusion_matrix/src/closest_one.cpp . . . 255
confusion_matrix/src/configuration.cpp . . . 256
confusion_matrix/src/confusion_matrix.cpp . . . 258
confusion_matrix/src/confusion_matrix_best_configs.cpp . . . 258
confusion_matrix/src/create_segmentation_results.cpp . . . 259
confusion_matrix/src/csv_file_info.cpp . . . 260
confusion_matrix/src/data_set_identifier.cpp . . . 260
confusion_matrix/src/distance.cpp . . . 262
confusion_matrix/src/distance_score.cpp . . . 262
confusion_matrix/src/imu.cpp . . . 263
confusion_matrix/src/interpolated_data_set_paths.cpp . . . 264
confusion_matrix/src/main.cpp . . . 212
confusion_matrix/src/manual_segmentation_point.cpp . . . 264
confusion_matrix/src/order_configurations_by_quality.cpp . . . 266
confusion_matrix/src/python_output.cpp . . . 266
confusion_matrix/src/segment.cpp . . . 267
confusion_matrix/src/split_string.cpp . . . 268
confusion_matrix/test/data_set_identifier_test.cpp . . . 269
confusion_matrix/test/interpolated_data_set_paths_test.cpp . . . 270
confusion_matrix/test/main.cpp . . . 214
confusion_matrix/test/manual_segmentation_point_test.cpp . . . 271
confusion_matrix/test/segment_test.cpp . . . 275
confusion_matrix/test/split_string_test.cpp . . . 276
counting/include/above_threshold.hpp . . . 278
counting/include/average_comparison_value_calculator.hpp . . . 279
counting/include/half_maximum_comparison_value_calculator.hpp . . . 279
counting/include/is_relevant.hpp . . . 280
counting/include/percentage_of.hpp . . . 281
counting/include/run_above_threshold.hpp . . . 282
counting/src/above_threshold.cpp . . . 283
counting/src/average_comparison_value_calculator.cpp . . . 285
counting/src/half_maximum_comparison_value_calculator.cpp . . . 286
counting/src/main.cpp . . . 206
counting/src/run_above_threshold.cpp . . . 286
counting/test/above_threshold_test.cpp . . . 287
counting/test/main.cpp . . . 208
counting/test/percentage_of_test.cpp . . . 289
csv_lib/include/cl/channel.hpp . . . 290
csv_lib/include/cl/column.hpp
Contains definitions regarding the columns of the 'old' CSV files, i.e., the CSV files that have not been preprocessed using MATLAB . . . 293
csv_lib/include/cl/data_point.hpp . . . 295
csv_lib/include/cl/data_set.hpp . . . 296
csv_lib/include/cl/dos2unix.hpp . . . 297
csv_lib/include/cl/error.hpp . . . 298
csv_lib/include/cl/exception.hpp . . . 299
csv_lib/include/cl/process.hpp . . . 307
csv_lib/include/cl/read_csv_file.hpp . . . 307
csv_lib/include/cl/s2n.hpp . . . 308
csv_lib/include/cl/sensor.hpp . . . 309
csv_lib/include/cl/to_string.hpp . . . 311

csv_lib/include/cl/use_unbuffered_io.hpp . . . 312
csv_lib/include/cl/fs/directory_listing.hpp . . . 301
csv_lib/include/cl/fs/file.hpp . . . 302
csv_lib/include/cl/fs/file_stream.hpp . . . 302
csv_lib/include/cl/fs/path.hpp . . . 303
csv_lib/include/cl/fs/separator.hpp . . . 304
csv_lib/include/cl/fs/windows.hpp
Contains Microsoft Windows specific functions . . . 305
csv_lib/src/cl/channel.cpp . . . 312
csv_lib/src/cl/data_point.cpp . . . 314
csv_lib/src/cl/data_set.cpp . . . 317
csv_lib/src/cl/dos2unix.cpp . . . 317
csv_lib/src/cl/error.cpp . . . 318
csv_lib/src/cl/exception.cpp . . . 319
csv_lib/src/cl/process.cpp . . . 322
csv_lib/src/cl/read_csv_file.cpp . . . 323
csv_lib/src/cl/sensor.cpp . . . 324
csv_lib/src/cl/use_unbuffered_io.cpp . . . 325
csv_lib/src/cl/fs/directory_listing.cpp . . . 319
csv_lib/src/cl/fs/file.cpp . . . 320
csv_lib/src/cl/fs/file_stream.cpp . . . 320
csv_lib/src/cl/fs/path.cpp . . . 321
csv_lib/src/cl/fs/windows.cpp . . . 322
csv_lib/test/channel_test.cpp . . . 325
csv_lib/test/column_test.cpp . . . 327
csv_lib/test/data_point_test.cpp . . . 329
csv_lib/test/data_set_test.cpp . . . 331
csv_lib/test/directory_listing_test.cpp . . . 335
csv_lib/test/error_test.cpp . . . 337
csv_lib/test/exception_test.cpp . . . 339
csv_lib/test/main.cpp . . . 209
csv_lib/test/read_csv_file_test.cpp . . . 340
csv_lib/test/s2n_test.cpp . . . 341
csv_lib/test/sensor_test.cpp . . . 343
csv_lib/test/to_string_test.cpp . . . 344
fix_csv/include/adjust_hardware_timestamp.hpp . . . 345
fix_csv/include/convert_to_unix_line_endings.hpp . . . 346
fix_csv/include/create_backup_file.hpp . . . 347
fix_csv/include/delete_non_bosch_sensors.hpp . . . 348
fix_csv/include/delete_out_of_bounds_values.hpp . . . 349
fix_csv/include/remove_zeros_from_field.hpp . . . 350
fix_csv/include/restore_from_backup.hpp . . . 351
fix_csv/include/write_file.hpp . . . 352
fix_csv/src/adjust_hardware_timestamp.cpp . . . 353
fix_csv/src/convert_to_unix_line_endings.cpp . . . 354
fix_csv/src/create_backup_file.cpp . . . 355
fix_csv/src/delete_non_bosch_sensors.cpp . . . 355
fix_csv/src/delete_out_of_bounds_values.cpp . . . 356
fix_csv/src/main.cpp . . . 210
fix_csv/src/remove_zeros_from_field.cpp . . . 357
fix_csv/src/restore_from_backup.cpp . . . 357
fix_csv/src/write_file.cpp . . . 358
fix_csv/test/adjust_hardware_timestamp_test.cpp . . . 359
fix_csv/test/main.cpp . . . 211
fix_csv/test/remove_zeros_from_field_test.cpp . . . 361

Chapter 5

Namespace Documentation

5.1 cl Namespace Reference

Namespaces

- fs

Classes

- struct col_traits

  Column traits for the columns of the old, non-preprocessed CSV files. Includes the index (0 based) of the column and the C++ data type to be used for the cell contents in that column.
- struct data_set_accessor

  Maps a `Channel` to its associated accessor member function pointer of the `DataSet` type.
- class DataPoint

  Type to represent a data point in a data set.
- class DataSet
- class Error
- class Exception
- class Process

Typedefs

- template<Column Col>
  using column_type = typename col_traits< Col >::type

  Meta function for the type to use for the given Column.
- template<typename Ty >
  using Expected = tl::expected< Ty, Error >

Enumerations

- enum Channel : std::uint64_t { Channel::CL_CHANNEL_X, Channel::CL_CHANNEL }

  Scoped enum for the 6 sensor channels.
- enum Column : std::size_t {
  Column::Time, Column::HardwareTimestamp, Column::ExtractId, Column::Trigger, Column::AccelerometerX, Column::AccelerometerY, Column::AccelerometerZ, Column::GyroscopeX, Column::GyroscopeY, Column::GyroscopeZ, Column::SamplingRate }

  A scoped enum for the columns of the old, non-preprocessed CSV files.
- enum CsvFileKind { CsvFileKind::Raw, CsvFileKind::Fixed }
- enum Sensor : std::uint64_t { Sensor::CL_SENSOR_X, Sensor::CL_SENSOR }

Functions

- DataSet::ChannelAccessor dataSetAccessor (Channel channel)

  Returns the data set accessor for the `Channel` given.
- std::ostream & operator<< (std::ostream &os, Channel channel)

  Prints a given `Channel` to `os`.
- bool isAccelerometer (Channel channel)

  Checks whether a given `Channel` is an accelerometer channel.
- bool isGyroscope (Channel channel)

  Checks whether a given `Channel` is a gyroscope channel.
- long double threshold (Channel channel)

  Returns the threshold value for `channel`.
- CL_SPECIALIZE_COL_TRAITS (Column::Time, long double)
- CL_SPECIALIZE_COL_TRAITS (Column::HardwareTimestamp, std::uint64_t)
- CL_SPECIALIZE_COL_TRAITS (Column::ExtractId, Sensor)
- CL_SPECIALIZE_COL_TRAITS (Column::Trigger, std::uint64_t)
- CL_SPECIALIZE_COL_TRAITS (Column::AccelerometerX, long double)
- CL_SPECIALIZE_COL_TRAITS (Column::AccelerometerY, long double)
- CL_SPECIALIZE_COL_TRAITS (Column::AccelerometerZ, long double)
- CL_SPECIALIZE_COL_TRAITS (Column::GyroscopeX, long double)
- CL_SPECIALIZE_COL_TRAITS (Column::GyroscopeY, long double)
- CL_SPECIALIZE_COL_TRAITS (Column::GyroscopeZ, long double)
- CL_SPECIALIZE_COL_TRAITS (Column::SamplingRate, std::uint64_t)
- std::vector< pl::byte > dos2unix (const void *p, std::size_t size)

  Converts DOS / Microsoft Windows line endings to UNIX line endings.
- Expected< std::vector< std::vector< std::string > > > readCsvFile (pl::string_view csvFilePath, std↩ ::vector< std::string > *columnNames=nullptr, CsvFileKind csvFileKind=CsvFileKind::Fixed) noexcept
- template<typename Integer >
  Expected< Integer > s2n (const std::string &str, std::size_t *pos=nullptr, [[maybe_unused]] int base=10)
- std::ostream & operator<< (std::ostream &os, Sensor sensor)
- template<typename Ty >
  std::string to_string (const Ty &ty)
- void useUnbufferedIo ()
- std::ostream & operator<< (std::ostream &os, const DataPoint &dataPoint)
- std::ostream & operator<< (std::ostream &os, const Error &error)

Variables

- constexpr std::size_t channelCount

  The amount of sensor channels (6).
- constexpr std::array< Channel, channelCount > channels

  An array of the sensor channel enumerators.
- template<Channel Chan>
  constexpr CL_CHANNEL DataSet::ChannelAccessor data_set_accessor_v = data_set_accessor<Chan>::f
- constexpr long double accelerometerThreshold {1.99L}

  Constant for the maximum acceptable accelerometer value (in positive and negative direction).
- constexpr long double gyroscopeThreshold {1999.99L}

  Constant for the maximum acceptable gyroscope value (in positive and negative direction).
- template<Column Col>
  constexpr std::size_t column_index = col_traits<Col>::index

  Meta function for the 0 based index of the given Column.
- constexpr std::array< Sensor, 4 > sensors

5.1.1 Typedef Documentation

5.1.1.1 column_type

```
template<Column Col>
using cl::column_type = typedef typename col_traits<Col>::type
```

Meta function for the type to use for the given Column.

Template Parameters

Col	The Column enumerator to use.

Definition at line 71 of file column.hpp.

5.1.1.2 Expected

```
template<typename Ty >
using cl::Expected = typedef tl::expected<Ty, Error>
```

Definition at line 64 of file error.hpp.

5.1.2 Enumeration Type Documentation

5.1.2.1 Channel

```
enum cl::Channel : std::uint64_t [strong]
```

Scoped enum for the 6 sensor channels.

Enumerator

CL_CHANNEL↩ _X	
CL_CHANNEL	

Definition at line 23 of file channel.hpp.

5.1.2.2 Column

```
enum cl::Column : std::size_t [strong]
```

A scoped enum for the columns of the old, non-preprocessed CSV files.

Enumerator

Time	
HardwareTimestamp	
ExtractId	
Trigger	
AccelerometerX	
AccelerometerY	
AccelerometerZ	
GyroscopeX	
GyroscopeY	
GyroscopeZ	
SamplingRate	

Definition at line 17 of file column.hpp.

5.1.2.3 CsvFileKind

```
enum cl::CsvFileKind [strong]
```

Enumerator

Raw	
Fixed	

Definition at line 11 of file read_csv_file.hpp.

5.1.2.4 Sensor

```
enum cl::Sensor : std::uint64_t [strong]
```

Enumerator

CL_SENSOR↩ _X	
CL_SENSOR	

Definition at line 15 of file sensor.hpp.

5.1.3 Function Documentation

5.1.3.1 CL_SPECIALIZE_COL_TRAITS() [1/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
            Column::AccelerometerX ,
            long double  )
```

5.1.3.2 CL_SPECIALIZE_COL_TRAITS() [2/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
            Column::AccelerometerY ,
            long double  )
```

5.1.3.3 CL_SPECIALIZE_COL_TRAITS() [3/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
            Column::AccelerometerZ ,
            long double  )
```

5.1.3.4 CL_SPECIALIZE_COL_TRAITS() [4/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
            Column::ExtractId ,
            Sensor  )
```

5.1.3.5 CL_SPECIALIZE_COL_TRAITS() [5/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
            Column::GyroscopeX ,
            long double  )
```

5.1.3.6 CL_SPECIALIZE_COL_TRAITS() [6/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
            Column::GyroscopeY ,
            long double  )
```

5.1.3.7 CL_SPECIALIZE_COL_TRAITS() [7/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
            Column::GyroscopeZ ,
            long double  )
```

5.1.3.8 CL_SPECIALIZE_COL_TRAITS() [8/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
            Column::HardwareTimestamp ,
            std::uint64_t  )
```

5.1.3.9 CL_SPECIALIZE_COL_TRAITS() [9/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
            Column::SamplingRate ,
            std::uint64_t  )
```

5.1.3.10 CL_SPECIALIZE_COL_TRAITS() [10/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
            Column::Time ,
            long double  )
```

5.1.3.11 CL_SPECIALIZE_COL_TRAITS() [11/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
            Column::Trigger ,
            std::uint64_t  )
```

5.1.3.12 dataSetAccessor()

```
DataSet::ChannelAccessor cl::dataSetAccessor (
            Channel channel )
```

Returns the data set accessor for the `Channel` given.

Parameters

channel	The `Channel` to get the data set accessor for.

Returns

The data set accessor for `channel`.

Definition at line 15 of file channel.cpp.

Here is the caller graph for this function:

5.1.3.13 dos2unix()

```
std::vector< pl::byte > cl::dos2unix (
            const void * p,
            std::size_t size )
```

Converts DOS / Microsoft Windows line endings to UNIX line endings.

Parameters

p	The beginning of the data to convert.
size	The size of the data to convert in bytes.

Returns

The resulting byte array.

Definition at line 4 of file dos2unix.cpp.

Here is the caller graph for this function:

5.1.3.14 isAccelerometer()

```
bool cl::isAccelerometer (
            Channel channel )
```

Checks whether a given `Channel` is an accelerometer channel.

Parameters

channel	The `Channel` to check.

Returns

true if `channel` is an accelerometer sensor channel; false otherwise.

Definition at line 45 of file channel.cpp.

Here is the caller graph for this function:

5.1.3.15 isGyroscope()

```
bool cl::isGyroscope (
            Channel channel )
```

Checks whether a given `Channel` is a gyroscope channel.

Parameters

channel	The `Channel` to check.

Returns

true if `channel` is a gyroscope sensor channel; false otherwise.

Definition at line 50 of file channel.cpp.

Here is the caller graph for this function:

5.1.3.16 operator<<() [1/4]

```
std::ostream & cl::operator<< (
            std::ostream & os,
            Channel channel )
```

Prints a given `Channel` to `os`.

Parameters

os	The ostream to print to.
channel	The `Channel` to print.

Returns

`os`.

Definition at line 32 of file channel.cpp.

5.1.3.17 operator<<() [2/4]

```
std::ostream& cl::operator<< (
            std::ostream & os,
            const DataPoint & dataPoint )
```

Parameters

os	The ostream to print to.
dataPoint	The `DataPoint` to print.

Returns

`os`

Definition at line 10 of file data_point.cpp.

Here is the call graph for this function:

5.1.3.18 operator<<() [3/4]

```
std::ostream& cl::operator<< (
            std::ostream & os,
            const Error & error )
```

Definition at line 30 of file error.cpp.

Here is the call graph for this function:

5.1.3.19 operator<<() [4/4]

```
std::ostream & cl::operator<< (
            std::ostream & os,
            Sensor sensor )
```

Definition at line 8 of file sensor.cpp.

Here is the call graph for this function:

5.1.3.20 readCsvFile()

```
Expected< std::vector< std::vector< std::string > > > cl::readCsvFile (
            pl::string_view csvFilePath,
            std::vector< std::string > * columnNames = nullptr,
            CsvFileKind csvFileKind = CsvFileKind::Fixed )  [noexcept]
```

Definition at line 50 of file read_csv_file.cpp.

Here is the caller graph for this function:

5.1.3.21 s2n()

```
template<typename Integer >
Expected<Integer> cl::s2n (
            const std::string & str,
            std::size_t * pos = nullptr,
            [[maybe_unused] ] int base = 10 )  [inline]
```

Definition at line 16 of file s2n.hpp.

5.1.3.22 threshold()

```
long double cl::threshold (
            Channel channel )
```

Returns the threshold value for `channel`.

Parameters

channel	The `Channel` to get the threshold value of.

Returns

The threshold value for `channel`.

Definition at line 55 of file channel.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.1.3.23 to_string()

```
template<typename Ty >
std::string cl::to_string (
            const Ty & ty )  [inline]
```

Definition at line 16 of file to_string.hpp.

Here is the caller graph for this function:

5.1.3.24 useUnbufferedIo()

```
void cl::useUnbufferedIo ( )
```

Definition at line 9 of file use_unbuffered_io.cpp.

Here is the caller graph for this function:

5.1.4 Variable Documentation

5.1.4.1 accelerometerThreshold

```
constexpr long double cl::accelerometerThreshold {1.99L}  [inline], [constexpr]
```

Constant for the maximum acceptable accelerometer value (in positive and negative direction).

Definition at line 102 of file channel.hpp.

5.1.4.2 channelCount

```
constexpr std::size_t cl::channelCount  [inline], [constexpr]
```

Initial value:

```
{0
#define CL_CHANNEL_X(enumerator, value, dataSetAccessor)
                                        CL_CHANNEL
}
```

The amount of sensor channels (6).

Definition at line 32 of file channel.hpp.

5.1.4.3 channels

```
constexpr std::array<Channel, channelCount> cl::channels  [inline], [constexpr]
```

Initial value:

```
{{
#define CL_CHANNEL_X(enm, v, a)
  CL_CHANNEL
}}
```

An array of the sensor channel enumerators.

Definition at line 41 of file channel.hpp.

5.1.4.4 column_index

```
template<Column Col>
constexpr std::size_t cl::column_index = col_traits<Col>::index  [inline], [constexpr]
```

Meta function for the 0 based index of the given Column.

Template Parameters

Col	The Column enumerator to use.

Definition at line 64 of file column.hpp.

5.1.4.5 data_set_accessor_v

```
template<Channel Chan>
constexpr CL_CHANNEL DataSet::ChannelAccessor cl::data_set_accessor_v = data_set_accessor<Chan>↩
::f  [inline], [constexpr]
```

Definition at line 65 of file channel.hpp.

5.1.4.6 gyroscopeThreshold

```
constexpr long double cl::gyroscopeThreshold {1999.99L}  [inline], [constexpr]
```

Constant for the maximum acceptable gyroscope value (in positive and negative direction).

Definition at line 108 of file channel.hpp.

5.1.4.7 sensors

```
constexpr std::array<Sensor, 4> cl::sensors  [inline], [constexpr]
```

Initial value:

```
{{
#define CL_SENSOR_X(enm, v)
  CL_SENSOR
}}
```

Definition at line 21 of file sensor.hpp.

5.2 cl::fs Namespace Reference

Classes

- class File

  Represents a file.
- class FileStream

  A binary file stream.
- class Path

  A filesystem path.

Enumerations

- enum DirectoryListingOption { DirectoryListingOption::None, DirectoryListingOption::ExcludeDotAndDotDot }

  Options for directoryListing.

Functions

- Expected< std::vector< Path > > directoryListing (const Path &directoryPath, DirectoryListingOption directoryListingOption=DirectoryListingOption::ExcludeDotAndDotDot)

  Creates a listing of the contents of a directory.
- std::wstring utf8ToUtf16 (pl::string_view utf8)

  Converts a UTF-8 encoded string to a UTF-16 encoded wstring.
- std::string utf16ToUtf8 (pl::wstring_view utf16)

  Converts a UTF-16 encoded wide character string to UTF-8 string.
- std::wstring formatError (DWORD errorCode)

  Formats a WINAPI error code to a UTF-16 encoded wide character string.
- std::ostream & operator<< (std::ostream &os, const Path &path)
- bool operator< (const Path &lhs, const Path &rhs) noexcept
- bool operator== (const Path &lhs, const Path &rhs) noexcept

5.2.1 Enumeration Type Documentation

5.2.1.1 DirectoryListingOption

```
enum cl::fs::DirectoryListingOption  [strong]
```

Options for directoryListing.

Enumerator

None	No option
ExcludeDotAndDotDot	Exclude the . and .. directories

Definition at line 13 of file directory_listing.hpp.

5.2.2 Function Documentation

5.2.2.1 directoryListing()

```
Expected< std::vector< Path > > cl::fs::directoryListing (
            const Path & directoryPath,
            DirectoryListingOption directoryListingOption = DirectoryListingOption::ExcludeDotAndDotDot
)
```

Creates a listing of the contents of a directory.

Parameters

directoryPath	The directory to list.
directoryListingOption	The option to use.

Returns

A vector of the directory entries or an error on failure.

Warning

The paths returned are not complete paths to the files, but merely the identifiers of the directory's contents.

Definition at line 24 of file directory_listing.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.2.2.2 formatError()

```
std::wstring cl::fs::formatError (
            DWORD errorCode )
```

Formats a WINAPI error code to a UTF-16 encoded wide character string.

Parameters

errorCode	The WINAPI error code.

Returns

The resulting UTF-16 encoded wide character string.

Note

Most WINAPIs expect UTF-16 encoded wide character strings, but we don't want to pollute the code base with UTF-16 strings.

Warning

Wide characters are only 16 bit wide on Microsoft Windows, they're 32 bit on GNU / Linux.

Definition at line 89 of file windows.cpp.

5.2.2.3 operator<()

```
bool cl::fs::operator< (
            const Path & lhs,
            const Path & rhs )  [noexcept]
```

Parameters

lhs	The left hand side operand.
rhs	The right hand side operand.

Returns

true if lhs < rhs; otherwise false.

Definition at line 27 of file path.cpp.

5.2.2.4 operator<<()

```
std::ostream& cl::fs::operator<< (
            std::ostream & os,
            const Path & path )
```

Parameters

os	the ostream to print to.
path	The path to print.

Returns

`os`

Definition at line 22 of file path.cpp.

5.2.2.5 operator==()

```
bool cl::fs::operator== (
            const Path & lhs,
            const Path & rhs )  [noexcept]
```

Parameters

lhs	The left hand side operand.
rhs	The right hand side operand.

Returns

true if `lhs` and `rhs` are equal.

Definition at line 32 of file path.cpp.

5.2.2.6 utf16ToUtf8()

```
std::string cl::fs::utf16ToUtf8 (
            pl::wstring_view utf16 )
```

Converts a UTF-16 encoded wide character string to UTF-8 string.

Parameters

utf16	The UTF-16 encoded wide character string to convert.

Returns

The resulting UTF-8 string.

Note

Most WINAPIs expect UTF-16 encoded wide character strings, but we don't want to pollute the code base with UTF-16 strings.

Warning

Wide characters are only 16 bit wide on Microsoft Windows, they're 32 bit on GNU / Linux.

Definition at line 61 of file windows.cpp.

Here is the caller graph for this function:

5.2.2.7 utf8ToUtf16()

```
std::wstring cl::fs::utf8ToUtf16 (
            pl::string_view utf8 )
```

Converts a UTF-8 encoded string to a UTF-16 encoded wstring.

Parameters

utf8	The UTF-8 encoded string to convert.

Returns

The resulting UTF-16 string.

Note

Most WINAPIs expect UTF-16 encoded wide character strings, but we don't want to pollute the code base with UTF-16 strings.

Warning

Wide characters are only 16 bit wide on Microsoft Windows, they're 32 bit on GNU / Linux.

Definition at line 35 of file windows.cpp.

Here is the caller graph for this function:

5.3 cm Namespace Reference

Classes

- class Configuration

  Represents a possible configuration for the Python segmentor.
- struct ConfigWithDistanceScore
- struct ConfigWithTotalConfusionMatrix

  A `Configuration` with a `ConfusionMatrix`.
- class ConfusionMatrix

  Type to represent a confusion matrix.
- class CsvFileInfo

  Type to hold the hardware timestamps of a CSV file.
- class ManualSegmentationPoint

  Type used to represent a manual segmentation point.

Enumerations

- enum DataSetIdentifier { DataSetIdentifier::CM_DATA_SET_IDENTIFIER_X, DataSetIdentifier::CM_DATA_SET_IDENTIFIER }

  Scoped enum type for the data set identifiers.
- enum Imu { Imu::CM_IMU_X, Imu::CM_IMU }

  Scoped enum type for the IMUs.

Functions

- std::uint64_t closestOne (std::uint64_t algorithmicallyDeterminedSegmentationPoint, const std::vector< std::uint64_t > &manualSegmentationPoints)

  Finds the segmentation point in `manualSegmentationPoints` that is the closest to `algorithmically↩ DeterminedSegmentationPoint`.
- CM_SORTER (truePositives, >)

  Sorts `ConfigWithTotalConfusionMatrix` objects by true positives (highest first)
- CM_SORTER (trueNegatives, >)

  Sorts `ConfigWithTotalConfusionMatrix` objects by true negatives (highest first)
- CM_SORTER (falsePositives,<)

  Sorts `ConfigWithTotalConfusionMatrix` objects by false positives (lowest first)
- CM_SORTER (falseNegatives,<)

  Sorts `ConfigWithTotalConfusionMatrix` objects by false negatives (lowest first)
- std::vector< ConfigWithTotalConfusionMatrix > confusionMatrixBestConfigs (const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t >> &manualSegmentationPoints, const std::unordered_↩ map< Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >>> &algorithmically↩ DeterminedSegmentationPoints, const std::function< bool(const ConfigWithTotalConfusionMatrix &, const ConfigWithTotalConfusionMatrix &)> &sorter)

  Determines the 'best' configurations.
- std::unordered_map< cm::Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > > createSegmentationResults ()

  Invokes Python to generate the segmentation points algorithmically.
- std::ostream & operator<< (std::ostream &os, DataSetIdentifier dsi)

  Prints a DataSetIdentifier to an ostream.
- DataSetIdentifier toDataSetIdentifier (const cl::fs::Path &path)

  Converts a path to a CSV file to the corresponding DataSetIdentifier.
- std::uint64_t distance (std::uint64_t a, std::uint64_t b)

  Calculates the distance between `a` and `b`.
- std::uint64_t distanceScore (const std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >> &segmentationPointsForConfig, const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t >> &manualSegmentationPoints)
- template<typename Map , typename Key >
  auto fetch (const Map &map, const Key &key)

  Fetches a value from a map for a given key.
- std::ostream & operator<< (std::ostream &os, Imu imu)

  Prints `imu` to `os`.
- std::vector< cl::fs::Path > interpolatedDataSetPaths ()

  Returns the paths to the interpolated data sets.
- bool operator< (const ConfigWithDistanceScore &lhs, const ConfigWithDistanceScore &rhs) noexcept
- std::ostream & operator<< (std::ostream &os, const ConfigWithDistanceScore &configWithDistScore)
- std::vector< ConfigWithDistanceScore > orderConfigurationsByQuality (const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t >> &manualSegmentationPoints, const std::unordered_↩ map< Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >>> &algorithmically↩ DeterminedSegmentationPoints)
- std::string pythonOutput (const cl::fs::Path &csvFilePath, const Configuration &segmentorConfiguration)

  Runs the Python segmentor on `path`.
- std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > segment (const Configuration &segmentorConfiguration)
- std::vector< std::string > splitString (std::string string, pl::string_view splitBy)

  Splits `string` by `splitBy`.
- bool operator== (const Configuration &lhs, const Configuration &rhs) noexcept
- bool operator< (const Configuration &lhs, const Configuration &rhs) noexcept

- std::ostream & operator<< (std::ostream &os, const Configuration &config)
- std::ostream & operator<< (std::ostream &os, const ConfigWithTotalConfusionMatrix &obj)
- bool operator== (const ManualSegmentationPoint &lhs, const ManualSegmentationPoint &rhs) noexcept
- bool operator!= (const ManualSegmentationPoint &lhs, const ManualSegmentationPoint &rhs) noexcept
- std::ostream & operator<< (std::ostream &os, const ManualSegmentationPoint &manualSegmentationPoint)

Variables

- struct {
  } disregardTrueNegativesSorter

  Sorter to sort `ConfigWithTotalConfusionMatrix` objects by the count of true positives minus the count of false positives minus the count of false negatives.
- struct {
  } addTrueSubtractFalseSorter

  Sorter to sort `ConfigWithTotalConfusionMatrix` objects by the sum of true positives and true negatives minus the false positives minus the false negatives.
- constexpr std::size_t imuCount

  The amount of IMUs.
- constexpr std::array< Imu, imuCount > imus

  An array of the IMU enumerators.

5.3.1 Enumeration Type Documentation

5.3.1.1 DataSetIdentifier

```
enum cm::DataSetIdentifier [strong]
```

Scoped enum type for the data set identifiers.

Enumerator

CM_DATA_SET_IDENTIFIER↩ _X	
CM_DATA_SET_IDENTIFIER	

Definition at line 33 of file data_set_identifier.hpp.

5.3.1.2 Imu

```
enum cm::Imu [strong]
```

Scoped enum type for the IMUs.

Enumerator

CM_IMU↩ _X	
CM_IMU	

Definition at line 17 of file imu.hpp.

5.3.2 Function Documentation

5.3.2.1 closestOne()

```
std::uint64_t cm::closestOne (
            std::uint64_t algorithmicallyDeterminedSegmentationPoint,
            const std::vector< std::uint64_t > & manualSegmentationPoints )
```

Finds the segmentation point in `manualSegmentationPoints` that is the closest to `algorithmically↩ DeterminedSegmentationPoint`.

Parameters

algorithmicallyDeterminedSegmentationPoint	The segmentation point to find the closest one to.
manualSegmentationPoints	The manual segmentation points.

Returns

The manual segmentation point that the the closest to `algorithmicallyDeterminedSegmentation↩ Point`.

Exceptions

cl::Exception	if no segmentation point was found.

Definition at line 11 of file closest_one.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.3.2.2 CM_SORTER() [1/4]

```
cm::CM_SORTER (
            falseNegatives  )
```

Sorts `ConfigWithTotalConfusionMatrix` objects by false negatives (lowest first)

5.3.2.3 CM_SORTER() [2/4]

```
cm::CM_SORTER (
            falsePositives  )
```

Sorts `ConfigWithTotalConfusionMatrix` objects by false positives (lowest first)

5.3.2.4 CM_SORTER() [3/4]

```
cm::CM_SORTER (
            trueNegatives  )
```

Sorts `ConfigWithTotalConfusionMatrix` objects by true negatives (highest first)

5.3.2.5 CM_SORTER() [4/4]

```
cm::CM_SORTER (
            truePositives  )
```

Sorts `ConfigWithTotalConfusionMatrix` objects by true positives (highest first)

5.3.2.6 confusionMatrixBestConfigs()

```
std::vector< ConfigWithTotalConfusionMatrix > cm::confusionMatrixBestConfigs (
            const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t >> &
manualSegmentationPoints,
            const std::unordered_map< Configuration, std::unordered_map< cl::fs::Path, std↩
::vector< std::uint64_t >>> & algorithmicallyDeterminedSegmentationPoints,
            const std::function< bool(const ConfigWithTotalConfusionMatrix &, const ConfigWithTotalConfusionM
&)> & sorter )
```

Determines the 'best' configurations.

Parameters

manualSegmentationPoints	The manual segmentation points (ground truth).
algorithmicallyDeterminedSegmentationPoints	The segmentation points found by the Python application.
sorter	The sorter to use.

Returns

A vector of `ConfigWithTotalConfusionMatrix` objects sorted by `sorter`.

Exceptions

cl::Exception	if `sorter` does not contain a valid target.

Definition at line 100 of file confusion_matrix_best_configs.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.3.2.7 createSegmentationResults()

```
std::unordered_map< Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64↩
_t > > > cm::createSegmentationResults ( )
```

Invokes Python to generate the segmentation points algorithmically.

Returns

A map that maps the configurations to maps that map CSV file paths to segmentation points.

Exceptions

cl::Exception	on error.

Definition at line 42 of file create_segmentation_results.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.3.2.8 distance()

```
std::uint64_t cm::distance (
            std::uint64_t a,
            std::uint64_t b )
```

Calculates the distance between a and b.

Parameters

a	The first parameter.
b	The second parameter.

Returns

The difference between a and b.

Definition at line 6 of file distance.cpp.

Here is the caller graph for this function:

5.3.2.9 distanceScore()

```
std::uint64_t cm::distanceScore (
            const std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >> & segmentation↩
PointsForConfig,
            const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t >> &
manualSegmentationPoints )
```

Definition at line 7 of file distance_score.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.3.2.10 fetch()

```
template<typename Map , typename Key >
auto cm::fetch (
            const Map & map,
            const Key & key )
```

Fetches a value from a map for a given key.

Template Parameters

Map	The type of the map.
Key	The type of the Key.

Parameters

map	The map to fetch from.
key	The key to find the value for in `map`.

Returns

The value for `key` in `map`.

Exceptions

cl::Exception	if `key` is not found in `map`.

Definition at line 16 of file fetch.hpp.

Here is the caller graph for this function:

5.3.2.11 interpolatedDataSetPaths()

```
std::vector< cl::fs::Path > cm::interpolatedDataSetPaths ( )
```

Returns the paths to the interpolated data sets.

Returns

The interpolated data set paths.

Exceptions

cl::Exception	on error.

Definition at line 62 of file interpolated_data_set_paths.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.3.2.12 operator"!=()

```
bool cm::operator!= (
            const ManualSegmentationPoint & lhs,
            const ManualSegmentationPoint & rhs )  [noexcept]
```

Parameters

lhs	The first operand.
rhs	The second operand.

Returns

true if `lhs` is considered not equal to `rhs`; false otherwise.

Definition at line 189 of file manual_segmentation_point.cpp.

5.3.2.13 operator<() [1/2]

```
bool cm::operator< (
            const Configuration & lhs,
            const Configuration & rhs )  [noexcept]
```

Parameters

lhs	The first operand.
rhs	The second operand.

Returns

true if `lhs` is considered less than `rhs`; otherwise false.

Definition at line 213 of file configuration.cpp.

5.3.2.14 operator<() [2/2]

```
bool cm::operator< (
            const ConfigWithDistanceScore & lhs,
            const ConfigWithDistanceScore & rhs )  [noexcept]
```

Definition at line 20 of file order_configurations_by_quality.cpp.

5.3.2.15 operator<<() [1/6]

```
std::ostream& cm::operator<< (
            std::ostream & os,
            const Configuration & config )
```

Parameters

os	The ostream to print to.
config	The `Configuration` to print.

Returns

os

Definition at line 233 of file configuration.cpp.

5.3.2.16 operator<<() [2/6]

```
std::ostream & cm::operator<< (
            std::ostream & os,
            const ConfigWithDistanceScore & configWithDistScore )
```

Definition at line 27 of file order_configurations_by_quality.cpp.

5.3.2.17 operator<<() [3/6]

```
std::ostream& cm::operator<< (
            std::ostream & os,
            const ConfigWithTotalConfusionMatrix & obj )
```

Parameters

os	The ostream to print to.
obj	The `ConfigWithTotalConfusionMatrix` to print.

Returns

os

Definition at line 69 of file confusion_matrix_best_configs.cpp.

Here is the call graph for this function:

5.3.2.18 operator<<() [4/6]

```
std::ostream& cm::operator<< (
            std::ostream & os,
            const ManualSegmentationPoint & manualSegmentationPoint )
```

Parameters

os	The ostream to print to
manualSegmentationPoint	The `ManualSegmentationPoint` to print.

Returns

os

Definition at line 196 of file manual_segmentation_point.cpp.

Here is the call graph for this function:

5.3.2.19 operator<<() [5/6]

```
std::ostream & cm::operator<< (
            std::ostream & os,
            DataSetIdentifier dsi )
```

Prints a DataSetIdentifier to an ostream.

Parameters

os	The ostream to print to.
dsi	The `DataSetIdentifier` to print.

Returns

`os`

Definition at line 33 of file data_set_identifier.cpp.

5.3.2.20 operator<<() [6/6]

```
std::ostream & cm::operator<< (
            std::ostream & os,
            Imu imu )
```

Prints `imu` to `os`.

Parameters

os	The ostream to print to
imu	The IMU enumerator to print.

Returns

`os`

Definition at line 36 of file imu.cpp.

5.3.2.21 operator==() [1/2]

```
bool cm::operator== (
            const Configuration & lhs,
            const Configuration & rhs )  [noexcept]
```

Parameters

lhs	The first operand.
rhs	The second operand.

Returns

true if `lhs` and `rhs` are considered to be equal.

Definition at line 193 of file configuration.cpp.

5.3.2.22 operator==() [2/2]

```
bool cm::operator== (
            const ManualSegmentationPoint & lhs,
            const ManualSegmentationPoint & rhs )  [noexcept]
```

Parameters

lhs	The first operand.
rhs	The second operand.

Returns

true if `lhs` is considered equal to `rhs`; false otherwise.

Definition at line 181 of file manual_segmentation_point.cpp.

5.3.2.23 orderConfigurationsByQuality()

```
std::vector< ConfigWithDistanceScore > cm::orderConfigurationsByQuality (
            const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t >> &
manualSegmentationPoints,
            const std::unordered_map< Configuration, std::unordered_map< cl::fs::Path, std↩
::vector< std::uint64_t >>> & algorithmicallyDeterminedSegmentationPoints )
```

Definition at line 38 of file order_configurations_by_quality.cpp.

Here is the call graph for this function:

5.3.2.24 pythonOutput()

```
std::string cm::pythonOutput (
            const cl::fs::Path & csvFilePath,
            const Configuration & segmentorConfiguration )
```

Runs the Python segmentor on `path`.

Parameters

path	The path to the CSV file to segment.
segmentorConfiguration	The configuration to use.

Returns

The output of the Python application.

Exceptions

cl::Exception	if creating the process failed.

Definition at line 32 of file python_output.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.3.2.25 segment()

```
std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > cm::segment (
            const Configuration & segmentorConfiguration )
```

Invokes Python to segment the interpolated data sets.

Parameters

segmentorConfiguration	The `Configuration` to use for the Python segmentor.

Returns

A map that maps the paths to the interpolated data sets to vectors of the hardware timestamps (in milliseconds) that are segmentation points.

Exceptions

cl::Exception	if an error occurs.

Definition at line 65 of file segment.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.3.2.26 splitString()

```
std::vector< std::string > cm::splitString (
            std::string string,
            pl::string_view splitBy )
```

Splits `string` by `splitBy`.

Parameters

string	The string to split.
splitBy	What to split `string` by.

Returns

The resulting strings.

Definition at line 8 of file split_string.cpp.

Here is the caller graph for this function:

5.3.2.27 toDataSetIdentifier()

```
DataSetIdentifier cm::toDataSetIdentifier (
            const cl::fs::Path & path )
```

Converts a path to a CSV file to the corresponding DataSetIdentifier.

Parameters

path	The path.

Returns

The resulting DataSetIdentifier.

Exceptions

cl::Exception	if `path` is unrecognized.

Definition at line 38 of file data_set_identifier.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.3.3 Variable Documentation

5.3.3.1 addTrueSubtractFalseSorter

```
constexpr { ... } cm::addTrueSubtractFalseSorter
```

Sorter to sort `ConfigWithTotalConfusionMatrix` objects by the sum of true positives and true negatives minus the false positives minus the false negatives.

5.3.3.2 disregardTrueNegativesSorter

```
constexpr { ... } cm::disregardTrueNegativesSorter
```

Sorter to sort `ConfigWithTotalConfusionMatrix` objects by the count of true positives minus the count of false positives minus the count of false negatives.

5.3.3.3 imuCount

```
constexpr std::size_t cm::imuCount  [inline], [constexpr]
```

Initial value:

```
{0
#define CM_IMU_X(enm)
                                        CM_IMU
}
```

The amount of IMUs.

Definition at line 26 of file imu.hpp.

5.3.3.4 imus

```
constexpr std::array<Imu, imuCount> cm::imus  [inline], [constexpr]
```

Initial value:

```
{{
#define CM_IMU_X(enm)
  CM_IMU
}}
```

An array of the IMU enumerators.

Definition at line 35 of file imu.hpp.

5.4 cs Namespace Reference

Classes

- class CsvLineBuilder

  Builder for a CSV line.
- struct data_set_info

  Meta function for data set tags.
- class LogInfo

  Information about a log file.
- class LogLine

  A line out of a log file.

Enumerations

- enum FilterKind { FilterKind::Butterworth, FilterKind::MovingAverage }

  Type for the different kinds of filters.
- enum Mode { Mode::CS_MODE_X, Mode::CS_MODE }

  Enumerator type for the different modes of the compare_segmentation application.
- enum SegmentationKind : pl::byte { SegmentationKind::Minima = 0b0000'0001, SegmentationKind::Maxima = 0b0000'0010, SegmentationKind::Both = Minima | Maxima }

  The segmentation kind.

Functions

- PL_DEFINE_EXCEPTION_TYPE (NoSuchDataSetException, std::logic_error)
- CS_SPECIALIZE_DATA_SET_INFO (Felix1, "11.17.39", 24)
- CS_SPECIALIZE_DATA_SET_INFO (Felix2, "12.50.00", 20)
- CS_SPECIALIZE_DATA_SET_INFO (Felix3, "13.00.09", 15)
- CS_SPECIALIZE_DATA_SET_INFO (Marcelle1, "14.59.59", 10)
- CS_SPECIALIZE_DATA_SET_INFO (Marcelle2, "15.13.22", 16)
- CS_SPECIALIZE_DATA_SET_INFO (Marcelle3, "15.31.36", 18)
- CS_SPECIALIZE_DATA_SET_INFO (Mike1, "14.07.33", 26)
- CS_SPECIALIZE_DATA_SET_INFO (Mike2, "14.14.32", 22)
- CS_SPECIALIZE_DATA_SET_INFO (Mike3, "14.20.28", 18)
- CS_SPECIALIZE_DATA_SET_INFO (Andre1, "Andre_liegestuetzen1", 27)
- CS_SPECIALIZE_DATA_SET_INFO (Andre2, "Andre_liegestuetzen2", 20)
- CS_SPECIALIZE_DATA_SET_INFO (Andre3, "Andre_liegestuetzen3", 17)
- CS_SPECIALIZE_DATA_SET_INFO (AndreSquats1, "Andre_Squats", 30)
- CS_SPECIALIZE_DATA_SET_INFO (AndreSquats2, "Andre_Squats2", 49)
- CS_SPECIALIZE_DATA_SET_INFO (Jan1, "Jan_liegestuetzen1", 25)
- CS_SPECIALIZE_DATA_SET_INFO (Jan2, "Jan_liegestuetzen2", 19)
- CS_SPECIALIZE_DATA_SET_INFO (Jan3, "Jan_liegestuetzen3", 13)
- CS_SPECIALIZE_DATA_SET_INFO (Lucas1, "Lukas_liegestuetzen1", 24)
- CS_SPECIALIZE_DATA_SET_INFO (Lucas2, "Lukas_liegestuetzen2", 19)
- CS_SPECIALIZE_DATA_SET_INFO (Lucas3, "Lukas_liegestuetzen3", 11)
- std::uint64_t repetitionCount (pl::string_view dataSet)

  Fetches the repetition count for a given data set identified by its string.
- std::ostream & operator<< (std::ostream &os, FilterKind filterKind)

  Prints a FilterKind to an ostream.
- cl::Expected< std::vector< cl::fs::Path > > logFiles (pl::string_view directoryPath)

  Fetches the paths to the log files in the given directory.
- std::ostream & operator<< (std::ostream &os, Mode mode)

  Prints a `Mode` to an ostream.
- cl::Expected< Mode > parseMode (const char *szCmdArg)

  Parses a null-terminated byte character string as a `Mode`.
- std::ostream & operator<< (std::ostream &os, SegmentationKind segmentationKind)

  Prints a SegmentationKind to an ostream.
- bool operator== (const LogInfo &lhs, const LogInfo &rhs) noexcept
- bool operator!= (const LogInfo &lhs, const LogInfo &rhs) noexcept
- std::ostream & operator<< (std::ostream &os, const LogInfo &logInfo)

Variables

- constexpr pl::string_view logPath {"segmentation_comparison/logs"}

  Relative path to the directory containing the preprocessed log files.
- constexpr pl::string_view oldLogPath {"segmentation_comparison/logs/old"}

  Relative path to the directory containing the old log files.

5.4.1 Enumeration Type Documentation

5.4.1.1 FilterKind

```
enum cs::FilterKind [strong]
```

Type for the different kinds of filters.

Enumerator

Butterworth	
MovingAverage	

Definition at line 9 of file filter_kind.hpp.

5.4.1.2 Mode

`enum cs::Mode [strong]`

Enumerator type for the different modes of the compare_segmentation application.

Enumerator

CS_MODE↩ _X	
CS_MODE	

Definition at line 19 of file mode.hpp.

5.4.1.3 SegmentationKind

`enum cs::SegmentationKind : pl::byte [strong]`

The segmentation kind.

Enumerator

Minima	Segmentation by local minima
Maxima	Segmentation by local maxima
Both	Segmentation by both local extrema

Definition at line 12 of file segmentation_kind.hpp.

5.4.2 Function Documentation

5.4.2.1 CS_SPECIALIZE_DATA_SET_INFO() [1/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Andre1 ,
```

```
            "Andre_liegestuetzen1" ,
            27  )
```

5.4.2.2 CS_SPECIALIZE_DATA_SET_INFO() [2/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Andre2 ,
            "Andre_liegestuetzen2" ,
            20  )
```

5.4.2.3 CS_SPECIALIZE_DATA_SET_INFO() [3/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Andre3 ,
            "Andre_liegestuetzen3" ,
            17  )
```

5.4.2.4 CS_SPECIALIZE_DATA_SET_INFO() [4/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            AndreSquats1 ,
            "Andre_Squats" ,
            30  )
```

5.4.2.5 CS_SPECIALIZE_DATA_SET_INFO() [5/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            AndreSquats2 ,
            "Andre_Squats2" ,
            49  )
```

5.4.2.6 CS_SPECIALIZE_DATA_SET_INFO() [6/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Felix1 ,
            "11.17.39" ,
            24  )
```

5.4.2.7 CS_SPECIALIZE_DATA_SET_INFO() [7/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Felix2 ,
            "12.50.00" ,
            20  )
```

5.4.2.8 CS_SPECIALIZE_DATA_SET_INFO() [8/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Felix3 ,
            "13.00.09" ,
            15  )
```

5.4.2.9 CS_SPECIALIZE_DATA_SET_INFO() [9/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Jan1 ,
            "Jan_liegestuetzen1" ,
            25  )
```

5.4.2.10 CS_SPECIALIZE_DATA_SET_INFO() [10/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Jan2 ,
            "Jan_liegestuetzen2" ,
            19  )
```

5.4.2.11 CS_SPECIALIZE_DATA_SET_INFO() [11/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Jan3 ,
            "Jan_liegestuetzen3" ,
            13  )
```

5.4.2.12 CS_SPECIALIZE_DATA_SET_INFO() [12/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Lucas1 ,
            "Lukas_liegestuetzen1" ,
            24  )
```

5.4.2.13 CS_SPECIALIZE_DATA_SET_INFO() [13/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Lucas2 ,
            "Lukas_liegestuetzen2" ,
            19  )
```

5.4.2.14 CS_SPECIALIZE_DATA_SET_INFO() [14/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Lucas3 ,
            "Lukas_liegestuetzen3" ,
            11  )
```

5.4.2.15 CS_SPECIALIZE_DATA_SET_INFO() [15/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Marcelle1 ,
            "14.59.59" ,
            10  )
```

5.4.2.16 CS_SPECIALIZE_DATA_SET_INFO() [16/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Marcelle2 ,
            "15.13.22" ,
            16  )
```

5.4.2.17 CS_SPECIALIZE_DATA_SET_INFO() [17/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Marcelle3 ,
            "15.31.36" ,
            18  )
```

5.4.2.18 CS_SPECIALIZE_DATA_SET_INFO() [18/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Mike1 ,
            "14.07.33" ,
            26  )
```

5.4.2.19 CS_SPECIALIZE_DATA_SET_INFO() [19/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Mike2 ,
            "14.14.32" ,
            22  )
```

5.4.2.20 CS_SPECIALIZE_DATA_SET_INFO() [20/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
            Mike3 ,
            "14.20.28" ,
            18  )
```

5.4.2.21 logFiles()

```
cl::Expected< std::vector< cl::fs::Path > > cs::logFiles (
            pl::string_view directoryPath )
```

Fetches the paths to the log files in the given directory.

Parameters

directoryPath	The path to a directory to search for log files.

Returns

The log files found or an error.

Definition at line 9 of file log_files.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.4.2.22 operator"!=()

```
bool cs::operator!= (
            const LogInfo & lhs,
            const LogInfo & rhs )  [noexcept]
```

Parameters

lhs	The left hand side operand.
rhs	The right hand side operand.

Returns

true if `lhs` and `rhs` are considered not equal; otherwise false.

Definition at line 287 of file log_info.cpp.

5.4.2.23 operator<<() [1/4]

```
std::ostream& cs::operator<< (
            std::ostream & os,
            const LogInfo & logInfo )
```

Parameters

os	The ostream to print to.
logInfo	The LogInfo to print.

Returns

os

Definition at line 292 of file log_info.cpp.

5.4.2.24 operator<<() [2/4]

```
std::ostream & cs::operator<< (
            std::ostream & os,
            FilterKind filterKind )
```

Prints a FilterKind to an ostream.

Parameters

os	The ostream to print to.
filterKind	The FilterKind to print.

Returns

os

Definition at line 6 of file filter_kind.cpp.

5.4.2.25 operator<<() [3/4]

```
std::ostream & cs::operator<< (
            std::ostream & os,
            Mode mode )
```

Prints a `Mode` to an ostream.

Parameters

os	The ostream to print to.
mode	The `Mode` to print.

Returns

os

Definition at line 13 of file mode.cpp.

5.4.2.26 operator<<() [4/4]

```
std::ostream & cs::operator<< (
            std::ostream & os,
            SegmentationKind segmentationKind )
```

Prints a SegmentationKind to an ostream.

Parameters

os	The ostream to print to.
segmentationKind	The SegmentationKind to print.

Returns

os

Definition at line 6 of file segmentation_kind.cpp.

5.4.2.27 operator==()

```
bool cs::operator== (
            const LogInfo & lhs,
            const LogInfo & rhs )  [noexcept]
```

Parameters

lhs	The left hand side operand.
rhs	The right hand side operand.

Returns

true if `lhs` and `rhs` are considered equal; otherwise false.

Definition at line 264 of file log_info.cpp.

5.4.2.28 parseMode()

```
cl::Expected< Mode > cs::parseMode (
            const char * szCmdArg )
```

Parses a null-terminated byte character string as a `Mode`.

Parameters

szCmdArg	A null-terminated byte character string containing a textual representation of a `Mode`.

Returns

The mode parsed out of `szCmdArg` or an error.

Definition at line 25 of file mode.cpp.

Here is the caller graph for this function:

5.4.2.29 PL_DEFINE_EXCEPTION_TYPE()

```
cs::PL_DEFINE_EXCEPTION_TYPE (
            NoSuchDataSetException ,
            std::logic_error  )
```

5.4.2.30 repetitionCount()

```
std::uint64_t cs::repetitionCount (
            pl::string_view dataSet )
```

Fetches the repetition count for a given data set identified by its string.

Parameters

dataSet	The data set to fetch the repetition count of.

Returns

The repetition count of `dataSet`.

Warning

`dataSet` may not be invalid!

Definition at line 10 of file data_set_info.cpp.

Here is the caller graph for this function:

5.4.3 Variable Documentation

5.4.3.1 logPath

```
constexpr pl::string_view cs::logPath {"segmentation_comparison/logs"}  [inline], [constexpr]
```

Relative path to the directory containing the preprocessed log files.

Definition at line 9 of file paths.hpp.

5.4.3.2 oldLogPath

```
constexpr pl::string_view cs::oldLogPath {"segmentation_comparison/logs/old"}  [inline], [constexpr]
```

Relative path to the directory containing the old log files.

Definition at line 14 of file paths.hpp.

5.5 ctg Namespace Reference

Functions

- std::vector< cl::DataPoint > aboveThreshold (const cl::DataSet &dataSet, long double accelerometer↩ Threshold, long double gyroscopeThreshold)
- long double averageComparisonValueCalculator (cl::Sensor sensor, cl::Channel channel, const cl::DataSet &dataSet)
- long double halfMaximumComparisonValueCalculator (cl::Sensor sensor, cl::Channel channel, const cl::DataSet &dataSet)
- template<typename ComparisonValueCalculator >
  bool isRelevant (cl::Sensor sensor, cl::Channel channel, const cl::DataSet &dataSet, ComparisonValue↩ Calculator comparisonValueCalculator)
- constexpr long double percentageOf (std::size_t amount, std::size_t totalCount) noexcept
- void runAboveThreshold (std::ostream &aboveThresholdLogOutputFileStream, const cl::DataSet &dataSet)

5.5.1 Function Documentation

5.5.1.1 aboveThreshold()

```
std::vector< cl::DataPoint > ctg::aboveThreshold (
            const cl::DataSet & dataSet,
            long double accelerometerThreshold,
            long double gyroscopeThreshold )
```

Definition at line 28 of file above_threshold.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.5.1.2 averageComparisonValueCalculator()

```
long double ctg::averageComparisonValueCalculator (
            cl::Sensor sensor,
            cl::Channel channel,
            const cl::DataSet & dataSet )
```

Definition at line 10 of file average_comparison_value_calculator.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.5.1.3 halfMaximumComparisonValueCalculator()

```
long double ctg::halfMaximumComparisonValueCalculator (
            cl::Sensor sensor,
            cl::Channel channel,
            const cl::DataSet & dataSet )
```

Definition at line 10 of file half_maximum_comparison_value_calculator.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.5.1.4 isRelevant()

```
template<typename ComparisonValueCalculator >
bool ctg::isRelevant (
            cl::Sensor sensor,
            cl::Channel channel,
            const cl::DataSet & dataSet,
            ComparisonValueCalculator comparisonValueCalculator )
```

Definition at line 11 of file is_relevant.hpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.5.1.5 percentageOf()

```
constexpr long double ctg::percentageOf (
            std::size_t amount,
            std::size_t totalCount )  [constexpr], [noexcept]
```

Definition at line 6 of file percentage_of.hpp.

Here is the caller graph for this function:

5.5.1.6 runAboveThreshold()

```
void ctg::runAboveThreshold (
            std::ostream & aboveThresholdLogOutputFileStream,
            const cl::DataSet & dataSet )
```

Definition at line 14 of file run_above_threshold.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.6 fmc Namespace Reference

Functions

- void adjustHardwareTimestamp (std::string ∗cellContent, const std::string &nextRowHardwareTimestamp, std::uint64_t ∗overflowCount)
- bool convertToUnixLineEndings (const std::string &csvPath)
- bool createBackupFile (const std::string &csvFilePath, const std::string &backupFilePath)
- void deleteNonBoschSensors (std::vector< std::vector< std::string >> ∗data)
- cl::Expected< void > deleteOutOfBoundsValues (std::vector< std::vector< std::string >> ∗data)
- void removeZerosFromField (std::string ∗field)
- bool restoreFromBackup (const std::string &csvFilePath, const std::string &backupFilePath)
- bool writeFile (pl::string_view csvPath, pl::string_view csvFileExtension, const std::vector< std::string > &columnNames, const std::vector< std::vector< std::string >> &data)

5.6.1 Function Documentation

5.6.1.1 adjustHardwareTimestamp()

```
void fmc::adjustHardwareTimestamp (
            std::string * cellContent,
            const std::string & nextRowHardwareTimestamp,
            std::uint64_t * overflowCount )
```

Definition at line 16 of file adjust_hardware_timestamp.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.6.1.2 convertToUnixLineEndings()

```
bool fmc::convertToUnixLineEndings (
            const std::string & csvPath )
```

Definition at line 18 of file convert_to_unix_line_endings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.6.1.3 createBackupFile()

```
bool fmc::createBackupFile (
            const std::string & csvFilePath,
            const std::string & backupFilePath )
```

Definition at line 6 of file create_backup_file.cpp.

Here is the caller graph for this function:

5.6.1.4 deleteNonBoschSensors()

```
void fmc::deleteNonBoschSensors (
            std::vector< std::vector< std::string >> * data )
```

Definition at line 30 of file delete_non_bosch_sensors.cpp.

Here is the caller graph for this function:

5.6.1.5 deleteOutOfBoundsValues()

```
cl::Expected< void > fmc::deleteOutOfBoundsValues (
            std::vector< std::vector< std::string >> * data )
```

Definition at line 29 of file delete_out_of_bounds_values.cpp.

Here is the caller graph for this function:

5.6.1.6 removeZerosFromField()

```
void fmc::removeZerosFromField (
            std::string * field )
```

Definition at line 6 of file remove_zeros_from_field.cpp.

Here is the caller graph for this function:

5.6.1.7 restoreFromBackup()

```
bool fmc::restoreFromBackup (
            const std::string & csvFilePath,
            const std::string & backupFilePath )
```

Definition at line 11 of file restore_from_backup.cpp.

Here is the caller graph for this function:

5.6.1.8 writeFile()

```
bool fmc::writeFile (
            pl::string_view csvPath,
            pl::string_view csvFileExtension,
            const std::vector< std::string > & columnNames,
            const std::vector< std::vector< std::string >> & data )
```

Definition at line 12 of file write_file.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

Chapter 6

Class Documentation

6.1 cm::Configuration::Builder Class Reference

Builder type for `Configuration`.

```
#include <configuration.hpp>
```

Public Member Functions

- Builder () noexcept

  Creates an empty `Builder`.
- Builder & skipWindow (bool value)

  Sets the skipWindow property.
- Builder & deleteTooClose (bool value)

  Sets the deleteTooClose property.
- Builder & deleteTooLowVariance (bool value)

  Sets the deleteTooLowVariance property.
- Builder & imu (Imu value)

  Sets the imu property.
- Builder & segmentationKind (std::string value)

  Sets the segmentationKind property.
- Builder & windowSize (std::size_t value)

  Sets the windowSize property.
- Builder & filterKind (std::string value)

  Sets the filterKind property.
- Configuration build () const

  Builds a `Configuration`.

6.1.1 Detailed Description

Builder type for `Configuration`.

Definition at line 41 of file configuration.hpp.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Builder()

`cm::Configuration::Builder::Builder ( ) [noexcept]`

Creates an empty `Builder`.

Definition at line 39 of file configuration.cpp.

6.1.3 Member Function Documentation

6.1.3.1 build()

`Configuration cm::Configuration::Builder::build ( ) const`

Builds a `Configuration`.

Returns

The `Configuration` built.

Exceptions

cl::Exception	if one of the properties has not been set or is invalid.

Definition at line 93 of file configuration.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.1.3.2 deleteTooClose()

```
Configuration::Builder & cm::Configuration::Builder::deleteTooClose (
            bool value )
```

Sets the deleteTooClose property.

Parameters

value	The value to use.

Returns

∗this

Definition at line 56 of file configuration.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.1.3.3 deleteTooLowVariance()

```
Configuration::Builder & cm::Configuration::Builder::deleteTooLowVariance (
            bool value )
```

Sets the deleteTooLowVariance property.

Parameters

value	The value to use.

Returns

*this

Definition at line 62 of file configuration.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.1.3.4 filterKind()

```
Configuration::Builder & cm::Configuration::Builder::filterKind (
            std::string value )
```

Sets the filterKind property.

Parameters

value	The value to use.

Returns

*this

Definition at line 87 of file configuration.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.1.3.5 imu()

```
Configuration::Builder & cm::Configuration::Builder::imu (
            Imu value )
```

Sets the imu property.

Parameters

value	The value to use.

Returns

∗this

Definition at line 68 of file configuration.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.1.3.6 segmentationKind()

```
Configuration::Builder & cm::Configuration::Builder::segmentationKind (
            std::string value )
```

Sets the segmentationKind property.

Parameters

value	The value to use.

Returns

∗this

Definition at line 74 of file configuration.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.1.3.7 skipWindow()

```
Configuration::Builder & cm::Configuration::Builder::skipWindow (
            bool value )
```

Sets the skipWindow property.

Parameters

value	The value to use.

Returns

*this

Definition at line 50 of file configuration.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.1.3.8 windowSize()

```
Configuration::Builder & cm::Configuration::Builder::windowSize (
            std::size_t value )
```

Sets the windowSize property.

Parameters

value	The value to use.

Returns

∗this

Definition at line 81 of file configuration.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

- confusion_matrix/include/configuration.hpp
- confusion_matrix/src/configuration.cpp

6.2 cl::col_traits< Col > Struct Template Reference

Column traits for the columns of the old, non-preprocessed CSV files. Includes the index (0 based) of the column and the C++ data type to be used for the cell contents in that column.

```
#include <column.hpp>
```

6.2.1 Detailed Description

template<Column Col>
struct cl::col_traits< Col >

Column traits for the columns of the old, non-preprocessed CSV files. Includes the index (0 based) of the column and the C++ data type to be used for the cell contents in that column.

Template Parameters

Col	The Column enumerator.

Definition at line 38 of file column.hpp.

The documentation for this struct was generated from the following file:

- csv_lib/include/cl/column.hpp

6.3 cm::Configuration Class Reference

Represents a possible configuration for the Python segmentor.

`#include <configuration.hpp>`

Classes

- class Builder

  Builder type for `Configuration`.

Public Member Functions

- Configuration ()

  Default constructor.
- bool skipWindow () const noexcept

  Read accessor for the skipWindow property.
- bool deleteTooClose () const noexcept

  Read accessor for the deleteTooClose property.
- bool deleteTooLowVariance () const noexcept

  Read accessor for the deleteTooLowVariance property.
- Imu imu () const noexcept

  Read accessor for the imu property.
- const std::string & segmentationKind () const noexcept

  Read accessor for the segmentationKind property.
- std::size_t windowSize () const noexcept

  Read accessor for the windowSize property.
- const std::string & filterKind () const noexcept

  Read accessor for the filterKind property.
- bool isInitialized () const noexcept

  Checks if this object is initialized and thus valid.
- cl::fs::Path createFilePath () const

  Create a file path for this kind of `Configuration`.
- bool serializeSegmentationPoints (const std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >> &segmentationPointsMap) const

  Serializes a map of segmentation points to the file path for this `Configuration`.
- std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > importSegmentationPoints () const

  Imports segmentation points from the file path for this `Configuration`.

Static Public Member Functions

- static const std::deque< bool > & skipWindowOptions () noexcept

  Returns the possible skipWindow options.
- static const std::deque< bool > & deleteTooCloseOptions () noexcept

  Returns the possible deleteTooClose options.
- static const std::deque< bool > & deleteTooLowVarianceOptions () noexcept

  Returns the possible deleteTooLowVariance options.
- static const std::vector< Imu > & imuOptions () noexcept

  Returns the possible imu options.
- static const std::vector< std::string > & segmentationKindOptions () noexcept

  Returns the possible segmentationKind options.
- static const std::vector< std::size_t > & windowSizeOptions () noexcept

  Returns the possible windowSize options.
- static const std::vector< std::string > & filterKindOptions () noexcept

  Returns the possible filterKind options.

Friends

- class Builder
- struct std::hash< Configuration >
- bool operator== (const Configuration &lhs, const Configuration &rhs) noexcept

  Compares two Configurations for equality.
- bool operator< (const Configuration &lhs, const Configuration &rhs) noexcept

  Less than compares two Configurations.
- std::ostream & operator<< (std::ostream &os, const Configuration &config)

  Prints `config` to `os`.

6.3.1 Detailed Description

Represents a possible configuration for the Python segmentor.

Definition at line 33 of file configuration.hpp.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Configuration()

`cm::Configuration::Configuration ( )`

Default constructor.

Warning

This constructor is only there to work around Microsoft buggedness, don't use.

Note

Creates an uninitialized object!

Definition at line 254 of file configuration.cpp.

6.3.3 Member Function Documentation

6.3.3.1 createFilePath()

```
cl::fs::Path cm::Configuration::createFilePath ( ) const
```

Create a file path for this kind of `Configuration`.

Returns

The file path for this kind of `Configuration`.

Definition at line 291 of file configuration.cpp.

Here is the caller graph for this function:

6.3.3.2 deleteTooClose()

```
bool cm::Configuration::deleteTooClose ( ) const  [noexcept]
```

Read accessor for the deleteTooClose property.

Returns

The deleteTooClose option.

Definition at line 268 of file configuration.cpp.

Here is the caller graph for this function:

6.3.3.3 deleteTooCloseOptions()

```
const std::deque< bool > & cm::Configuration::deleteTooCloseOptions ( ) [static], [noexcept]
```

Returns the possible deleteTooClose options.

Returns

The deleteTooClose options.

Definition at line 154 of file configuration.cpp.

Here is the caller graph for this function:

6.3.3.4 deleteTooLowVariance()

```
bool cm::Configuration::deleteTooLowVariance ( ) const [noexcept]
```

Read accessor for the deleteTooLowVariance property.

Returns

The deleteTooLowVariance option.

Definition at line 270 of file configuration.cpp.

Here is the caller graph for this function:

6.3.3.5 deleteTooLowVarianceOptions()

```
const std::deque< bool > & cm::Configuration::deleteTooLowVarianceOptions ( ) [static], [noexcept]
```

Returns the possible deleteTooLowVariance options.

Returns

The deleteTooLowVariance options.

Definition at line 160 of file configuration.cpp.

Here is the caller graph for this function:

6.3.3.6 filterKind()

```
const std::string & cm::Configuration::filterKind ( ) const [noexcept]
```

Read accessor for the filterKind property.

Returns

The filterKind option.

Definition at line 284 of file configuration.cpp.

Here is the caller graph for this function:

6.3.3.7 filterKindOptions()

```
const std::vector< std::string > & cm::Configuration::filterKindOptions ( ) [static], [noexcept]
```

Returns the possible filterKind options.

Returns

The filterKind options.

Definition at line 187 of file configuration.cpp.

Here is the caller graph for this function:

6.3.3.8 importSegmentationPoints()

```
std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > cm::Configuration::import↩
SegmentationPoints ( ) const
```

Imports segmentation points from the file path for this `Configuration`.

Returns

The imported segmentation points.

Exceptions

cl::Exception	if the file path for this `Configuration` does not exist or an error occurs while reading / parsing.

Definition at line 330 of file configuration.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.3.3.9 imu()

```
Imu cm::Configuration::imu ( ) const  [noexcept]
```

Read accessor for the imu property.

Returns

The imu option.

Definition at line 275 of file configuration.cpp.

Here is the caller graph for this function:

6.3.3.10 imuOptions()

```
const std::vector< Imu > & cm::Configuration::imuOptions ( ) [static], [noexcept]
```

Returns the possible imu options.

Returns

The imu options.

Definition at line 166 of file configuration.cpp.

Here is the caller graph for this function:

6.3.3.11 isInitialized()

```
bool cm::Configuration::isInitialized ( ) const [noexcept]
```

Checks if this object is initialized and thus valid.

Returns

true if this object is initialized; false otherwise.

Warning

If you use the Builder to construct (as you should) this member function will always return true. false will only be returned if you use the (invalid) default constructor that servers as a workaround for MSVC bugs.

Definition at line 289 of file configuration.cpp.

6.3.3.12 segmentationKind()

```
const std::string & cm::Configuration::segmentationKind ( ) const  [noexcept]
```

Read accessor for the segmentationKind property.

Returns

The segmentationKind option.

Definition at line 277 of file configuration.cpp.

Here is the caller graph for this function:

6.3.3.13 segmentationKindOptions()

```
const std::vector< std::string > & cm::Configuration::segmentationKindOptions ( )  [static],
[noexcept]
```

Returns the possible segmentationKind options.

Returns

The segmentationKind options.

Definition at line 173 of file configuration.cpp.

Here is the caller graph for this function:

6.3.3.14 serializeSegmentationPoints()

```
bool cm::Configuration::serializeSegmentationPoints (
            const std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >> & segmentation↩
PointsMap ) const
```

Serializes a map of segmentation points to the file path for this `Configuration`.

Parameters

segmentationPointsMap	The map to serialize.

Returns

true on success; false otherwise.

Definition at line 309 of file configuration.cpp.

Here is the call graph for this function:

6.3.3.15 skipWindow()

```
bool cm::Configuration::skipWindow ( ) const  [noexcept]
```

Read accessor for the skipWindow property.

Returns

The skipWindow option.

Definition at line 266 of file configuration.cpp.

Here is the caller graph for this function:

6.3.3.16 skipWindowOptions()

```
const std::deque< bool > & cm::Configuration::skipWindowOptions ( ) [static], [noexcept]
```

Returns the possible skipWindow options.

Returns

The skipWindow options.

Definition at line 148 of file configuration.cpp.

Here is the caller graph for this function:

6.3.3.17 windowSize()

```
std::size_t cm::Configuration::windowSize ( ) const [noexcept]
```

Read accessor for the windowSize property.

Returns

The windowSize option.

Definition at line 282 of file configuration.cpp.

Here is the caller graph for this function:

6.3.3.18 windowSizeOptions()

```
const std::vector< std::size_t > & cm::Configuration::windowSizeOptions ( ) [static], [noexcept]
```

Returns the possible windowSize options.

Returns

The windowSize options.

Definition at line 179 of file configuration.cpp.

Here is the caller graph for this function:

6.3.4 Friends And Related Function Documentation

6.3.4.1 Builder

```
friend class Builder [friend]
```

Definition at line 35 of file configuration.hpp.

6.3.4.2 operator<

```
bool operator< (
            const Configuration & lhs,
            const Configuration & rhs ) [friend]
```

Less than compares two Configurations.

Parameters

lhs	The first operand.
rhs	The second operand.

Returns

true if `lhs` is considered less than `rhs`; otherwise false.

Definition at line 213 of file configuration.cpp.

6.3.4.3 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const Configuration & config )  [friend]
```

Prints `config` to `os`.

Parameters

os	The ostream to print to.
config	The `Configuration` to print.

Returns

`os`

Definition at line 233 of file configuration.cpp.

6.3.4.4 operator==

```
bool operator== (
            const Configuration & lhs,
            const Configuration & rhs )  [friend]
```

Compares two Configurations for equality.

Parameters

lhs	The first operand.
rhs	The second operand.

Returns

true if `lhs` and `rhs` are considered to be equal.

Definition at line 193 of file configuration.cpp.

6.3.4.5 std::hash< Configuration >

```
friend struct std::hash< Configuration >  [friend]
```

Definition at line 36 of file configuration.hpp.

The documentation for this class was generated from the following files:

- confusion_matrix/include/configuration.hpp
- confusion_matrix/src/configuration.cpp

6.4 cm::ConfigWithDistanceScore Struct Reference

```
#include <order_configurations_by_quality.hpp>
```

Collaboration diagram for cm::ConfigWithDistanceScore:

Public Member Functions

- ConfigWithDistanceScore (Configuration p_config, std::uint64_t p_distScore)

Public Attributes

- Configuration config
- std::uint64_t distScore

6.4.1 Detailed Description

Definition at line 15 of file order_configurations_by_quality.hpp.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 ConfigWithDistanceScore()

```
cm::ConfigWithDistanceScore::ConfigWithDistanceScore (
            Configuration p_config,
            std::uint64_t p_distScore )
```

Definition at line 13 of file order_configurations_by_quality.cpp.

6.4.3 Member Data Documentation

6.4.3.1 config

```
Configuration cm::ConfigWithDistanceScore::config
```

Definition at line 18 of file order_configurations_by_quality.hpp.

6.4.3.2 distScore

```
std::uint64_t cm::ConfigWithDistanceScore::distScore
```

Definition at line 19 of file order_configurations_by_quality.hpp.

The documentation for this struct was generated from the following files:

- confusion_matrix/include/order_configurations_by_quality.hpp
- confusion_matrix/src/order_configurations_by_quality.cpp

6.5 cm::ConfigWithTotalConfusionMatrix Struct Reference

A `Configuration` with a `ConfusionMatrix`.

```
#include <confusion_matrix_best_configs.hpp>
```

Collaboration diagram for cm::ConfigWithTotalConfusionMatrix:

Public Member Functions

- ConfigWithTotalConfusionMatrix ()=default

  Default constructor.
- ConfigWithTotalConfusionMatrix (Configuration p_config, ConfusionMatrix p_matrix)

  Constructor.

Public Attributes

- Configuration config
- ConfusionMatrix matrix

Friends

- std::ostream & operator<< (std::ostream &os, const ConfigWithTotalConfusionMatrix &obj)

  Prints a `ConfigWithTotalConfusionMatrix` to `os`.

6.5.1 Detailed Description

A `Configuration` with a `ConfusionMatrix`.

Definition at line 16 of file confusion_matrix_best_configs.hpp.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 ConfigWithTotalConfusionMatrix() [1/2]

```
cm::ConfigWithTotalConfusionMatrix::ConfigWithTotalConfusionMatrix ( ) [default]
```

Default constructor.

6.5.2.2 ConfigWithTotalConfusionMatrix() [2/2]

```
cm::ConfigWithTotalConfusionMatrix::ConfigWithTotalConfusionMatrix (
            Configuration p_config,
            ConfusionMatrix p_matrix )
```

Constructor.

Parameters

p_config	The `Configuration` to use.
p_matrix	The `ConfusionMatrix` to use.

Definition at line 93 of file confusion_matrix_best_configs.cpp.

6.5.3 Friends And Related Function Documentation

6.5.3.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const ConfigWithTotalConfusionMatrix & obj )  [friend]
```

Prints a `ConfigWithTotalConfusionMatrix` to `os`.

Parameters

os	The ostream to print to.
obj	The `ConfigWithTotalConfusionMatrix` to print.

Returns

`os`

Definition at line 69 of file confusion_matrix_best_configs.cpp.

6.5.4 Member Data Documentation

6.5.4.1 config

```
Configuration cm::ConfigWithTotalConfusionMatrix::config
```

The `Configuration`

Definition at line 41 of file confusion_matrix_best_configs.hpp.

6.5.4.2 matrix

```
ConfusionMatrix cm::ConfigWithTotalConfusionMatrix::matrix
```

The associated `ConfusionMatrix`

Definition at line 42 of file confusion_matrix_best_configs.hpp.

The documentation for this struct was generated from the following files:

- confusion_matrix/include/confusion_matrix_best_configs.hpp
- confusion_matrix/src/confusion_matrix_best_configs.cpp

6.6 cm::ConfusionMatrix Class Reference

Type to represent a confusion matrix.

```
#include <confusion_matrix.hpp>
```

Public Types

- using this_type = ConfusionMatrix

Public Member Functions

- ConfusionMatrix ()

  Default constructs a `ConfusionMatrix` initializing all data members with 0.
- std::uint64_t truePositives () const noexcept

  Read accessor for the true positive count.
- std::uint64_t trueNegatives () const noexcept

  Read accessor for the true negative count.
- std::uint64_t falsePositives () const noexcept

  Read accessor for the false positive count.
- std::uint64_t falseNegatives () const noexcept

  Read accessor for the false negative count.
- std::uint64_t totalCount () const noexcept

  Read accessor for the total count.
- this_type & incrementTruePositives () noexcept

  Increments the true positive count and the total count.
- this_type & incrementTrueNegatives () noexcept

  Increments the true negative count and the total count.
- this_type & incrementFalsePositives () noexcept

  Increments the false positive count and the total count.
- this_type & incrementFalseNegatives () noexcept

  Increments the false negative count and the total count.
- this_type & operator+= (const ConfusionMatrix &other) noexcept

  Accumulates `other` into ∗this.

6.6.1 Detailed Description

Type to represent a confusion matrix.

Definition at line 9 of file confusion_matrix.hpp.

6.6.2 Member Typedef Documentation

6.6.2.1 this_type

```
using cm::ConfusionMatrix::this_type = ConfusionMatrix
```

Definition at line 11 of file confusion_matrix.hpp.

6.6.3 Constructor & Destructor Documentation

6.6.3.1 ConfusionMatrix()

```
cm::ConfusionMatrix::ConfusionMatrix ( )
```

Default constructs a `ConfusionMatrix` initializing all data members with 0.

Definition at line 4 of file confusion_matrix.cpp.

6.6.4 Member Function Documentation

6.6.4.1 falseNegatives()

```
std::uint64_t cm::ConfusionMatrix::falseNegatives ( ) const  [noexcept]
```

Read accessor for the false negative count.

Returns

The false negative count.

Definition at line 28 of file confusion_matrix.cpp.

6.6.4.2 falsePositives()

```
std::uint64_t cm::ConfusionMatrix::falsePositives ( ) const  [noexcept]
```

Read accessor for the false positive count.

Returns

The false positive count.

Definition at line 23 of file confusion_matrix.cpp.

6.6.4.3 incrementFalseNegatives()

```
ConfusionMatrix & cm::ConfusionMatrix::incrementFalseNegatives ( ) [noexcept]
```

Increments the false negative count and the total count.

Returns

*this

Definition at line 59 of file confusion_matrix.cpp.

Here is the caller graph for this function:

6.6.4.4 incrementFalsePositives()

```
ConfusionMatrix & cm::ConfusionMatrix::incrementFalsePositives ( ) [noexcept]
```

Increments the false positive count and the total count.

Returns

*this

Definition at line 52 of file confusion_matrix.cpp.

Here is the caller graph for this function:

6.6.4.5 incrementTrueNegatives()

```
ConfusionMatrix & cm::ConfusionMatrix::incrementTrueNegatives ( ) [noexcept]
```

Increments the true negative count and the total count.

Returns

*this

Definition at line 45 of file confusion_matrix.cpp.

Here is the caller graph for this function:

6.6.4.6 incrementTruePositives()

```
ConfusionMatrix & cm::ConfusionMatrix::incrementTruePositives ( ) [noexcept]
```

Increments the true positive count and the total count.

Returns

*this

Definition at line 38 of file confusion_matrix.cpp.

Here is the caller graph for this function:

6.6.4.7 operator+=()

```
ConfusionMatrix & cm::ConfusionMatrix::operator+= (
            const ConfusionMatrix & other ) [noexcept]
```

Accumulates `other` into *this.

Parameters

other	The other `ConfusionMatrix` to add to this `ConfusionMatrix`.

Returns

∗this

Definition at line 66 of file confusion_matrix.cpp.

6.6.4.8 totalCount()

```
std::uint64_t cm::ConfusionMatrix::totalCount ( ) const  [noexcept]
```

Read accessor for the total count.

Returns

The total count.

Definition at line 33 of file confusion_matrix.cpp.

Here is the caller graph for this function:

6.6.4.9 trueNegatives()

```
std::uint64_t cm::ConfusionMatrix::trueNegatives ( ) const  [noexcept]
```

Read accessor for the true negative count.

Returns

The true negative count.

Definition at line 18 of file confusion_matrix.cpp.

6.6.4.10 truePositives()

```
std::uint64_t cm::ConfusionMatrix::truePositives ( ) const  [noexcept]
```

Read accessor for the true positive count.

Returns

The true positive count.

Definition at line 13 of file confusion_matrix.cpp.

The documentation for this class was generated from the following files:

- confusion_matrix/include/confusion_matrix.hpp
- confusion_matrix/src/confusion_matrix.cpp

6.7 cm::CsvFileInfo Class Reference

Type to hold the hardware timestamps of a CSV file.

```
#include <csv_file_info.hpp>
```

Public Member Functions

- CsvFileInfo (const cl::fs::Path &csvFilePath)

  Reads the hardware timestamps from `csvFilePath`.
- const std::vector< std::uint64_t > & hardwareTimestamps () const noexcept

  Read accessor for the hardware timestamps.

6.7.1 Detailed Description

Type to hold the hardware timestamps of a CSV file.

Definition at line 13 of file csv_file_info.hpp.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 CsvFileInfo()

```
cm::CsvFileInfo::CsvFileInfo (
            const cl::fs::Path & csvFilePath )  [explicit]
```

Reads the hardware timestamps from `csvFilePath`.

Parameters

csvFilePath	The CSV to read the hardware timestamps from.

Exceptions

cl::Exception	on error.

Definition at line 10 of file csv_file_info.cpp.

6.7.3 Member Function Documentation

6.7.3.1 hardwareTimestamps()

```
const std::vector< std::uint64_t > & cm::CsvFileInfo::hardwareTimestamps ( ) const  [noexcept]
```

Read accessor for the hardware timestamps.

Returns

The hardware timestamps.

Definition at line 57 of file csv_file_info.cpp.

The documentation for this class was generated from the following files:

- confusion_matrix/include/csv_file_info.hpp
- confusion_matrix/src/csv_file_info.cpp

6.8 cs::CsvLineBuilder Class Reference

Builder for a CSV line.

```
#include <csv_line.hpp>
```

Public Types

- using this_type = CsvLineBuilder

Public Member Functions

- CsvLineBuilder ()

  Creates an empty, invalid CsvLineBuilder.
- this_type & skipWindow (bool value)

  Write accessor for the skip window property.
- this_type & deleteTooClose (bool value)

  Write accessor for the delete too close property.
- this_type & deleteLowVariance (bool value)

  Write accessor for the delete low variance property.
- this_type & kind (SegmentationKind value)

  Write accessor for the kind property.
- this_type & windowSize (std::uint64_t value)

  Write accessor for the window size property.
- this_type & filter (FilterKind value)

  Write accessor for the filter property.
- this_type & dataSet (std::string value)

  Write accessor for the data set property.
- this_type & sensor (std::uint64_t value)

  Write accessor for the sensor property.
- this_type & repetitions (std::uint64_t value)

  Write accessor for the repetitions property.
- this_type & segmentationPoints (std::uint64_t value)

  Write accessor for the segmentation points property.
- this_type & isOld (bool value)

  Write accessor for the is old property.
- std::vector< std::string > build () const

  Builds the CSV line as a vector containing the cells of the CSV line.

6.8.1 Detailed Description

Builder for a CSV line.

Builder type for a CSV line. All write accessors have to be called before the build member function is called!

Definition at line 21 of file csv_line.hpp.

6.8.2 Member Typedef Documentation

6.8.2.1 this_type

```
using cs::CsvLineBuilder::this_type = CsvLineBuilder
```

Definition at line 23 of file csv_line.hpp.

6.8.3 Constructor & Destructor Documentation

6.8.3.1 CsvLineBuilder()

`cs::CsvLineBuilder::CsvLineBuilder ( )`

Creates an empty, invalid CsvLineBuilder.

Definition at line 44 of file csv_line.cpp.

6.8.4 Member Function Documentation

6.8.4.1 build()

`std::vector< std::string > cs::CsvLineBuilder::build ( ) const`

Builds the CSV line as a vector containing the cells of the CSV line.

Returns

The resulting vector of strings.

Warning

May only be called after all the write accessors have been called.

Definition at line 124 of file csv_line.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.4.2 dataSet()

```
CsvLineBuilder & cs::CsvLineBuilder::dataSet (
            std::string value )
```

Write accessor for the data set property.

Parameters

value	The value to use.

Returns

∗this

Definition at line 94 of file csv_line.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.4.3 deleteLowVariance()

```
CsvLineBuilder & cs::CsvLineBuilder::deleteLowVariance (
            bool value )
```

Write accessor for the delete low variance property.

Parameters

value	The value to use.

Returns

*this

Definition at line 70 of file csv_line.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.4.4 deleteTooClose()

```
CsvLineBuilder & cs::CsvLineBuilder::deleteTooClose (
            bool value )
```

Write accessor for the delete too close property.

Parameters

value	The value to use.

Returns

*this

Definition at line 64 of file csv_line.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.4.5 filter()

```
CsvLineBuilder & cs::CsvLineBuilder::filter (
            FilterKind value )
```

Write accessor for the filter property.

Parameters

value	The value to use.

Returns

*this

Definition at line 88 of file csv_line.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.4.6 isOld()

```
CsvLineBuilder & cs::CsvLineBuilder::isOld (
            bool value )
```

Write accessor for the is old property.

Parameters

value	The value to use.

Returns

*this

Definition at line 118 of file csv_line.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.4.7 kind()

```
CsvLineBuilder & cs::CsvLineBuilder::kind (
            SegmentationKind value )
```

Write accessor for the kind property.

Parameters

value	The value to use.

Returns

*this

Definition at line 76 of file csv_line.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.4.8 repetitions()

```
CsvLineBuilder & cs::CsvLineBuilder::repetitions (
            std::uint64_t value )
```

Write accessor for the repetitions property.

Parameters

value	The value to use.

Returns

*this

Definition at line 106 of file csv_line.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.4.9 segmentationPoints()

```
CsvLineBuilder & cs::CsvLineBuilder::segmentationPoints (
            std::uint64_t value )
```

Write accessor for the segmentation points property.

Parameters

value	The value to use.

Returns

*this

Definition at line 112 of file csv_line.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.4.10 sensor()

```
CsvLineBuilder & cs::CsvLineBuilder::sensor (
            std::uint64_t value )
```

Write accessor for the sensor property.

Parameters

value	The value to use.

Returns

*this

Definition at line 100 of file csv_line.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.4.11 skipWindow()

```
CsvLineBuilder & cs::CsvLineBuilder::skipWindow (
            bool value )
```

Write accessor for the skip window property.

Parameters

value	The value to use.

Returns

*this

Definition at line 58 of file csv_line.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.4.12 windowSize()

```
CsvLineBuilder & cs::CsvLineBuilder::windowSize (
            std::uint64_t value )
```

Write accessor for the window size property.

Parameters

value	The value to use.

Returns

*this

Definition at line 82 of file csv_line.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

- compare_segmentation/include/csv_line.hpp
- compare_segmentation/src/csv_line.cpp

6.9 cl::data_set_accessor< Chan > Struct Template Reference

Maps a `Channel` to its associated accessor member function pointer of the `DataSet` type.

```
#include <channel.hpp>
```

6.9.1 Detailed Description

template<Channel Chan>
struct cl::data_set_accessor< Chan >

Maps a `Channel` to its associated accessor member function pointer of the `DataSet` type.

Template Parameters

Chan	The `Channel` to map to its data set accessor.

Definition at line 53 of file channel.hpp.

The documentation for this struct was generated from the following file:

- csv_lib/include/cl/channel.hpp

6.10 cs::data_set_info< Tag > Struct Template Reference

Meta function for data set tags.

```
#include <data_set_info.hpp>
```

6.10.1 Detailed Description

template<typename Tag>
struct cs::data_set_info< Tag >

Meta function for data set tags.

Template Parameters

Tag	The data set tag to use.

Meta function for data set tags. Contains a text for the data set tag and its repetition count.

Definition at line 21 of file data_set_info.hpp.

The documentation for this struct was generated from the following file:

- compare_segmentation/include/data_set_info.hpp

6.11 cl::DataPoint Class Reference

Type to represent a data point in a data set.

```
#include <data_point.hpp>
```

Public Member Functions

- DataPoint (std::string fileName, long double time, Sensor sensor, Channel channel, long double value) noexcept

  Creates a `DataPoint`.
- const std::string & fileName () const noexcept

  Read accessor for the file name.
- long double time () const noexcept
- Sensor sensor () const noexcept

  Read accessor for the sensor.
- Channel channel () const noexcept

  Read accessor for the channel.
- long double value () const noexcept

  Read accessor for the `DataPoint` value.

Friends

- std::ostream & operator<< (std::ostream &os, const DataPoint &dataPoint)

  Prints a `DataPoint` to `os`.

6.11.1 Detailed Description

Type to represent a data point in a data set.

Definition at line 13 of file data_point.hpp.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 DataPoint()

```
DataPoint::DataPoint (
            std::string fileName,
            long double time,
            Sensor sensor,
            Channel channel,
            long double value )  [noexcept]
```

Creates a `DataPoint`.

Parameters

fileName	The file name of the CSV file that contains this `DataPoint`.
time	The time.
sensor	The sensor.
channel	The channel.
value	The value.

Definition at line 21 of file data_point.cpp.

Here is the call graph for this function:

6.11.3 Member Function Documentation

6.11.3.1 channel()

```
Channel DataPoint::channel ( ) const  [noexcept]
```

Read accessor for the channel.

Returns

The channel.

Definition at line 41 of file data_point.cpp.

Here is the caller graph for this function:

6.11.3.2 fileName()

```
const std::string & DataPoint::fileName ( ) const  [noexcept]
```

Read accessor for the file name.

Returns

The file name.

Definition at line 35 of file data_point.cpp.

Here is the caller graph for this function:

6.11.3.3 sensor()

```
Sensor DataPoint::sensor ( ) const  [noexcept]
```

Read accessor for the sensor.

Returns

The sensor.

Definition at line 39 of file data_point.cpp.

Here is the caller graph for this function:

6.11.3.4 time()

```
long double DataPoint::time ( ) const  [noexcept]
```

\brif Read accessor for the time.

Returns

The time.

Definition at line 37 of file data_point.cpp.

Here is the caller graph for this function:

6.11.3.5 value()

```
long double DataPoint::value ( ) const  [noexcept]
```

Read accessor for the `DataPoint` value.

Returns

The value.

Definition at line 43 of file data_point.cpp.

Here is the caller graph for this function:

6.11.4 Friends And Related Function Documentation

6.11.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const DataPoint & dataPoint )  [friend]
```

Prints a DataPoint to os.

Parameters

os	The ostream to print to.
dataPoint	The DataPoint to print.

Returns

os

Definition at line 10 of file data_point.cpp.

The documentation for this class was generated from the following files:

- csv_lib/include/cl/data_point.hpp
- csv_lib/src/cl/data_point.cpp

6.12 cl::DataSet Class Reference

```
#include <data_set.hpp>
```

Public Types

- using size_type = std::size_t
- using ChannelAccessor = long double(DataSet::*)(size_type) const

Public Member Functions

- size_type rowCount () const noexcept
- const std::string & fileName () const noexcept
- column_type< Column::Time > time (size_type index) const
- column_type< Column::HardwareTimestamp > hardwareTimestamp (size_type index) const
- column_type< Column::ExtractId > extractId (size_type index) const
- column_type< Column::Trigger > trigger (size_type index) const
- column_type< Column::AccelerometerX > accelerometerX (size_type index) const
- column_type< Column::AccelerometerY > accelerometerY (size_type index) const
- column_type< Column::AccelerometerZ > accelerometerZ (size_type index) const
- column_type< Column::GyroscopeX > gyroscopeX (size_type index) const
- column_type< Column::GyroscopeY > gyroscopeY (size_type index) const
- column_type< Column::GyroscopeZ > gyroscopeZ (size_type index) const
- long double accelerometerAverage (Sensor sensor) const
- long double gyroscopeAverage (Sensor sensor) const
- long double accelerometerMaximum (Sensor sensor) const
- long double gyroscopeMaximum (Sensor sensor) const

Static Public Member Functions

- static Expected< DataSet > create (std::string fileName, const std::vector< std::vector< std::string >> &matrix)

6.12.1 Detailed Description

Definition at line 14 of file data_set.hpp.

6.12.2 Member Typedef Documentation

6.12.2.1 ChannelAccessor

```
using cl::DataSet::ChannelAccessor = long double (DataSet::*)(size_type) const
```

Definition at line 17 of file data_set.hpp.

6.12.2.2 size_type

```
using cl::DataSet::size_type = std::size_t
```

Definition at line 16 of file data_set.hpp.

6.12.3 Member Function Documentation

6.12.3.1 accelerometerAverage()

```
long double cl::DataSet::accelerometerAverage (
            Sensor sensor ) const
```

Definition at line 255 of file data_set.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.12.3.2 accelerometerMaximum()

```
long double cl::DataSet::accelerometerMaximum (
            Sensor sensor ) const
```

Definition at line 265 of file data_set.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.12.3.3 accelerometerX()

```
column_type< Column::AccelerometerX > cl::DataSet::accelerometerX (
            size_type index ) const
```

Definition at line 200 of file data_set.cpp.

Here is the caller graph for this function:

6.12.3.4 accelerometerY()

```
column_type< Column::AccelerometerY > cl::DataSet::accelerometerY (
            size_type index ) const
```

Definition at line 208 of file data_set.cpp.

Here is the caller graph for this function:

6.12.3.5 accelerometerZ()

```
column_type< Column::AccelerometerZ > cl::DataSet::accelerometerZ (
            size_type index ) const
```

Definition at line 216 of file data_set.cpp.

Here is the caller graph for this function:

6.12.3.6 create()

```
Expected< DataSet > cl::DataSet::create (
            std::string fileName,
            const std::vector< std::vector< std::string >> & matrix )  [static]
```

Definition at line 42 of file data_set.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.12.3.7 extractId()

```
column_type< Column::ExtractId > cl::DataSet::extractId (
            size_type index ) const
```

Definition at line 186 of file data_set.cpp.

Here is the caller graph for this function:

6.12.3.8 fileName()

```
const std::string & cl::DataSet::fileName ( ) const  [noexcept]
```

Definition at line 169 of file data_set.cpp.

Here is the caller graph for this function:

6.12.3.9 gyroscopeAverage()

```
long double cl::DataSet::gyroscopeAverage (
            Sensor sensor ) const
```

Definition at line 260 of file data_set.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.12.3.10 gyroscopeMaximum()

```
long double cl::DataSet::gyroscopeMaximum (
            Sensor sensor ) const
```

Definition at line 270 of file data_set.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.12.3.11 gyroscopeX()

```
column_type< Column::GyroscopeX > cl::DataSet::gyroscopeX (
            size_type index ) const
```

Definition at line 224 of file data_set.cpp.

Here is the caller graph for this function:

6.12.3.12 gyroscopeY()

```
column_type< Column::GyroscopeY > cl::DataSet::gyroscopeY (
            size_type index ) const
```

Definition at line 231 of file data_set.cpp.

Here is the caller graph for this function:

6.12.3.13 gyroscopeZ()

```
column_type< Column::GyroscopeZ > cl::DataSet::gyroscopeZ (
            size_type index ) const
```

Definition at line 238 of file data_set.cpp.

Here is the caller graph for this function:

6.12.3.14 hardwareTimestamp()

```
column_type< Column::HardwareTimestamp > cl::DataSet::hardwareTimestamp (
            size_type index ) const
```

Definition at line 178 of file data_set.cpp.

Here is the caller graph for this function:

6.12.3.15 rowCount()

```
DataSet::size_type cl::DataSet::rowCount ( ) const  [noexcept]
```

Definition at line 152 of file data_set.cpp.

Here is the caller graph for this function:

6.12.3.16 time()

```
column_type< Column::Time > cl::DataSet::time (
            size_type index ) const
```

Definition at line 171 of file data_set.cpp.

Here is the caller graph for this function:

6.12.3.17 trigger()

```
column_type< Column::Trigger > cl::DataSet::trigger (
            size_type index ) const
```

Definition at line 193 of file data_set.cpp.

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

- csv_lib/include/cl/data_set.hpp
- csv_lib/src/cl/data_set.cpp

6.13 cl::Error Class Reference

```
#include <error.hpp>
```

Public Types

- enum Kind { CL_ERROR_KIND }

Public Member Functions

- Error (Kind kind, std::string file, std::string function, std::size_t line, std::string message)
- Kind kind () const noexcept
- const std::string & file () const noexcept
- const std::string & function () const noexcept
- std::size_t line () const noexcept
- const std::string & message () const noexcept
- void raise () const
- std::string to_string () const

Friends

- std::ostream & operator<< (std::ostream &os, const Error &error)

6.13.1 Detailed Description

Definition at line 23 of file error.hpp.

6.13.2 Member Enumeration Documentation

6.13.2.1 Kind

```
enum cl::Error::Kind
```

Enumerator

CL_ERROR_KIND	

Definition at line 26 of file error.hpp.

6.13.3 Constructor & Destructor Documentation

6.13.3.1 Error()

```
cl::Error::Error (
            Kind kind,
            std::string file,
            std::string function,
            std::size_t line,
            std::string message )
```

Definition at line 41 of file error.cpp.

6.13.4 Member Function Documentation

6.13.4.1 file()

```
const std::string & cl::Error::file ( ) const  [noexcept]
```

Definition at line 57 of file error.cpp.

Here is the caller graph for this function:

6.13.4.2 function()

```
const std::string & cl::Error::function ( ) const  [noexcept]
```

Definition at line 59 of file error.cpp.

Here is the caller graph for this function:

6.13.4.3 kind()

```
Error::Kind cl::Error::kind ( ) const  [noexcept]
```

Definition at line 55 of file error.cpp.

Here is the caller graph for this function:

6.13.4.4 line()

```
std::size_t cl::Error::line ( ) const  [noexcept]
```

Definition at line 61 of file error.cpp.

6.13.4.5 message()

```
const std::string & cl::Error::message ( ) const  [noexcept]
```

Definition at line 63 of file error.cpp.

Here is the caller graph for this function:

6.13.4.6 raise()

```
void cl::Error::raise ( ) const
```

Definition at line 65 of file error.cpp.

6.13.4.7 to_string()

```
std::string cl::Error::to_string ( ) const
```

Definition at line 74 of file error.cpp.

6.13.5 Friends And Related Function Documentation

6.13.5.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const Error & error )  [friend]
```

Definition at line 30 of file error.cpp.

The documentation for this class was generated from the following files:

- csv_lib/include/cl/error.hpp
- csv_lib/src/cl/error.cpp

6.14 cl::Exception Class Reference

```
#include <exception.hpp>
```

Inheritance diagram for cl::Exception:

Collaboration diagram for cl::Exception:

Public Types

- using base_type = std::runtime_error

Public Member Functions

- Exception (std::string file, std::string function, std::size_t line, const std::string &what_arg)
- Exception (std::string file, std::string function, std::size_t line, const char *what_arg)
- const std::string & file () const noexcept
- const std::string & function () const noexcept
- std::size_t line () const noexcept

6.14.1 Detailed Description

Definition at line 14 of file exception.hpp.

6.14.2 Member Typedef Documentation

6.14.2.1 base_type

```
using cl::Exception::base_type = std::runtime_error
```

Definition at line 16 of file exception.hpp.

6.14.3 Constructor & Destructor Documentation

6.14.3.1 Exception() [1/2]

```
cl::Exception::Exception (
            std::string file,
            std::string function,
            std::size_t line,
            const std::string & what_arg )
```

Definition at line 6 of file exception.cpp.

6.14.3.2 Exception() [2/2]

```
cl::Exception::Exception (
            std::string file,
            std::string function,
            std::size_t line,
            const char * what_arg )
```

Definition at line 18 of file exception.cpp.

6.14.4 Member Function Documentation

6.14.4.1 file()

```
const std::string & cl::Exception::file ( ) const  [noexcept]
```

Definition at line 30 of file exception.cpp.

Here is the caller graph for this function:

6.14.4.2 function()

```
const std::string & cl::Exception::function ( ) const  [noexcept]
```

Definition at line 32 of file exception.cpp.

Here is the caller graph for this function:

6.14.4.3 line()

```
std::size_t cl::Exception::line ( ) const  [noexcept]
```

Definition at line 34 of file exception.cpp.

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

- csv_lib/include/cl/exception.hpp
- csv_lib/src/cl/exception.cpp

6.15 cl::fs::File Class Reference

Represents a file.

```
#include <file.hpp>
```

Public Member Functions

- File (Path path)

  Creates a File from the given `path`.
- bool exists () const noexcept

  Determines if this file exists.
- bool create () const noexcept

  Creates this file.
- bool copyTo (const Path &copyToPath) const noexcept

  Copies this file in the filesystem.
- bool moveTo (const Path &newPath)

  Moves this file in the filesystem.
- bool remove () noexcept

  Deletes this file.
- std::int64_t size () const noexcept

  Determines the size of this file in bytes.
- const Path & path () const noexcept

  Read accessor for the path of this file.

6.15.1 Detailed Description

Represents a file.

Definition at line 11 of file file.hpp.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 File()

```
cl::fs::File::File (
            Path path )  [explicit]
```

Creates a File from the given `path`.

Parameters

path	The path to use.

Definition at line 21 of file file.cpp.

Here is the call graph for this function:

6.15.3 Member Function Documentation

6.15.3.1 copyTo()

```
bool cl::fs::File::copyTo (
            const Path & copyToPath ) const  [noexcept]
```

Copies this file in the filesystem.

Parameters

copyToPath	The path to copy to.

Returns

true if the file was successfully copied to `copyToPath`; otherwise false.

Warning

There should be no file that already exists at `copyToPath`.

Definition at line 56 of file file.cpp.

Here is the call graph for this function:

6.15.3.2 create()

```
bool cl::fs::File::create ( ) const  [noexcept]
```

Creates this file.

Returns

true if the file was successfully created; otherwise false.

Note

Will fail if the file already exists.

Definition at line 25 of file file.cpp.

Here is the call graph for this function:

6.15.3.3 exists()

```
bool cl::fs::File::exists ( ) const  [noexcept]
```

Determines if this file exists.

Returns

true if the file exists; otherwise false.

Definition at line 23 of file file.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.15.3.4 moveTo()

```
bool cl::fs::File::moveTo (
            const Path & newPath )
```

Moves this file in the filesystem.

Parameters

newPath	The path to move this file to.

Returns

true if the file was successfully moved to `newPath`; otherwise false.

Definition at line 100 of file file.cpp.

Here is the call graph for this function:

6.15.3.5 path()

```
const Path & cl::fs::File::path ( ) const  [noexcept]
```

Read accessor for the path of this file.

Returns

The path of this file.

Definition at line 169 of file file.cpp.

Here is the caller graph for this function:

6.15.3.6 remove()

```
bool cl::fs::File::remove ( ) [noexcept]
```

Deletes this file.

Returns

true if deleting succeeded; otherwise false.

Definition at line 117 of file file.cpp.

Here is the call graph for this function:

6.15.3.7 size()

```
std::int64_t cl::fs::File::size ( ) const [noexcept]
```

Determines the size of this file in bytes.

Returns

The size of this file in bytes or -1 on error.

Warning

Returns -1 on error.

Definition at line 128 of file file.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

- csv_lib/include/cl/fs/file.hpp
- csv_lib/src/cl/fs/file.cpp

6.16 cl::fs::FileStream Class Reference

A binary file stream.

```
#include <file_stream.hpp>
```

Public Types

- enum OpenMode : std::uint8_t { Read = 0b0000'0001, Write = 0b0000'0010, ReadWrite = Read | Write }

  The file open mode.
- using this_type = FileStream

Public Member Functions

- PL_NONCOPYABLE (FileStream)
- FileStream (this_type &&other) noexcept

  Move constructs from `other`.
- this_type & operator= (this_type &&other) noexcept

  Move assigns `other` to this file stream.
- ~FileStream ()

  Closes this file stream.
- bool write (const void *data, std::size_t byteCount)

  Writes data to the file.
- std::vector< pl::byte > readAll () const

  Reads the entire file into RAM.

Static Public Member Functions

- static Expected< FileStream > create (const File &file, OpenMode openMode)

  Creates a file stream.

6.16.1 Detailed Description

A binary file stream.

Definition at line 19 of file file_stream.hpp.

6.16.2 Member Typedef Documentation

6.16.2.1 this_type

```
using cl::fs::FileStream::this_type = FileStream
```

Definition at line 30 of file file_stream.hpp.

6.16.3 Member Enumeration Documentation

6.16.3.1 OpenMode

```
enum cl::fs::FileStream::OpenMode : std::uint8_t
```

The file open mode.

Enumerator

Read	Read only access
Write	Write only access
ReadWrite	Read and write access

Definition at line 24 of file file_stream.hpp.

6.16.4 Constructor & Destructor Documentation

6.16.4.1 FileStream()

```
cl::fs::FileStream::FileStream (
            this_type && other )  [noexcept]
```

Move constructs from `other`.

Parameters

other	The file stream to move construct from.

Definition at line 70 of file file_stream.cpp.

6.16.4.2 ~FileStream()

```
cl::fs::FileStream::~FileStream ( )
```

Closes this file stream.

Definition at line 84 of file file_stream.cpp.

6.16.5 Member Function Documentation

6.16.5.1 create()

```
Expected< FileStream > cl::fs::FileStream::create (
            const File & file,
            OpenMode openMode )  [static]
```

Creates a file stream.

Parameters

file	The file to open.
openMode	The open mode to use.

Returns

The file stream or an error.

Definition at line 36 of file file_stream.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.16.5.2 operator=()

```
FileStream & cl::fs::FileStream::operator= (
            this_type && other )  [noexcept]
```

Move assigns `other` to this file stream.

Parameters

other	The file stream to move assign to this file stream.

Returns

*this

Definition at line 77 of file file_stream.cpp.

6.16.5.3 PL_NONCOPYABLE()

```
cl::fs::FileStream::PL_NONCOPYABLE (
            FileStream  )
```

6.16.5.4 readAll()

```
std::vector< pl::byte > cl::fs::FileStream::readAll ( ) const
```

Reads the entire file into RAM.

Returns

The bytes read.

Definition at line 103 of file file_stream.cpp.

6.16.5.5 write()

```
bool cl::fs::FileStream::write (
            const void * data,
            std::size_t byteCount )
```

Writes data to the file.

Parameters

data	Pointer to the beginning of the memory region to write.
byteCount	The amount of bytes to write, starting from `data`.

Returns

true on success; otherwise false.

Definition at line 96 of file file_stream.cpp.

The documentation for this class was generated from the following files:

- csv_lib/include/cl/fs/file_stream.hpp
- csv_lib/src/cl/fs/file_stream.cpp

6.17 std::hash<::cl::fs::Path > Struct Reference

```
#include <path.hpp>
```

Public Member Functions

- size_t operator() (const ::cl::fs::Path &path) const

6.17.1 Detailed Description

Definition at line 90 of file path.hpp.

6.17.2 Member Function Documentation

6.17.2.1 operator()()

```
size_t std::hash<::cl::fs::Path >::operator() (
            const ::cl::fs::Path & path ) const  [inline]
```

Definition at line 91 of file path.hpp.

The documentation for this struct was generated from the following file:

- csv_lib/include/cl/fs/path.hpp

6.18 std::hash<::cm::Configuration > Struct Reference

```
#include <configuration.hpp>
```

Public Member Functions

- size_t operator() (const ::cm::Configuration &configuration) const

6.18.1 Detailed Description

Definition at line 299 of file configuration.hpp.

6.18.2 Member Function Documentation

6.18.2.1 operator()()

```
size_t std::hash<::cm::Configuration >::operator() (
            const ::cm::Configuration & configuration ) const  [inline]
```

Definition at line 300 of file configuration.hpp.

The documentation for this struct was generated from the following file:

- confusion_matrix/include/configuration.hpp

6.19 cs::LogInfo Class Reference

Information about a log file.

```
#include <log_info.hpp>
```

Public Member Functions

- LogInfo ()

  Creates an uninitialized LogInfo.
- const cl::fs::Path & logFilePath () const noexcept

  Read accessor for the log file path.
- bool skipWindow () const noexcept

  Read accessor for the skip window option.
- bool deleteTooClose () const noexcept

  Read accessor for the delete too close option.
- bool deleteLowVariance () const noexcept

  Read accessor for the delete low variance option.
- SegmentationKind segmentationKind () const noexcept

  Read accessor for the segmentation kind.
- std::uint64_t windowSize () const noexcept

  Read accessor for the window size.
- FilterKind filterKind () const noexcept

  Read accessor for the filter kind.
- std::uint64_t sensor () const noexcept

  Read accessor for the sensor.
- bool isInitialized () const noexcept

  Checks whether this LogInfo is initialized.

Static Public Member Functions

- static cl::Expected< LogInfo > create (cl::fs::Path logFilePath) noexcept

  Creates a LogInfo from the given log file path.

Static Public Attributes

- static const std::uint64_t invalidSensor = UINT64_C(0xFFFFFFFFFFFFFFFF)

  Represents an invalid sensor.

Friends

- bool operator== (const LogInfo &lhs, const LogInfo &rhs) noexcept

  Compares two LogInfos for equality.
- bool operator!= (const LogInfo &lhs, const LogInfo &rhs) noexcept

  Compares two LogInfos for inequality.
- std::ostream & operator<< (std::ostream &os, const LogInfo &logInfo)

  Prints a LogInfo to an ostream.

6.19.1 Detailed Description

Information about a log file.

Information about a log file that is extracted from the log file name.

Definition at line 20 of file log_info.hpp.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 LogInfo()

```
cs::LogInfo::LogInfo ( )
```

Creates an uninitialized LogInfo.

Warning

Should only be used in order to be assigned with an initialized LogInfo; otherwise use the create static member function.

Definition at line 304 of file log_info.cpp.

6.19.3 Member Function Documentation

6.19.3.1 create()

```
cl::Expected< LogInfo > cs::LogInfo::create (
            cl::fs::Path logFilePath )  [static], [noexcept]
```

Creates a LogInfo from the given log file path.

Parameters

logFilePath	The log file path to create a LogInfo from.

Returns

The LogInfo created or an error.

Definition at line 90 of file log_info.cpp.

Here is the caller graph for this function:

6.19.3.2 deleteLowVariance()

```
bool cs::LogInfo::deleteLowVariance ( ) const  [noexcept]
```

Read accessor for the delete low variance option.

Returns

true if delete low variance is active; false otherwise.

Definition at line 326 of file log_info.cpp.

6.19.3.3 deleteTooClose()

```
bool cs::LogInfo::deleteTooClose ( ) const  [noexcept]
```

Read accessor for the delete too close option.

Returns

true if delete too close is active; false otherwise.

Definition at line 324 of file log_info.cpp.

6.19.3.4 filterKind()

```
FilterKind cs::LogInfo::filterKind ( ) const  [noexcept]
```

Read accessor for the filter kind.

Returns

The filter kind.

Definition at line 335 of file log_info.cpp.

6.19.3.5 isInitialized()

```
bool cs::LogInfo::isInitialized ( ) const  [noexcept]
```

Checks whether this LogInfo is initialized.

Returns

true if this LogInfo is initialized; false otherwise.

Note

Will return true if this LogInfo was created with the create static member function.

Definition at line 339 of file log_info.cpp.

6.19.3.6 logFilePath()

```
const cl::fs::Path & cs::LogInfo::logFilePath ( ) const  [noexcept]
```

Read accessor for the log file path.

Returns

The log file path.

Definition at line 317 of file log_info.cpp.

Here is the caller graph for this function:

6.19.3.7 segmentationKind()

```
SegmentationKind cs::LogInfo::segmentationKind ( ) const  [noexcept]
```

Read accessor for the segmentation kind.

Returns

The segmentation kind.

Definition at line 328 of file log_info.cpp.

6.19.3.8 sensor()

```
std::uint64_t cs::LogInfo::sensor ( ) const  [noexcept]
```

Read accessor for the sensor.

Returns

The sensor.

Note

Will be the invalid sensor unless the log file is old.

Definition at line 337 of file log_info.cpp.

6.19.3.9 skipWindow()

```
bool cs::LogInfo::skipWindow ( ) const  [noexcept]
```

Read accessor for the skip window option.

Returns

true if skip window is active; false otherwise.

Definition at line 322 of file log_info.cpp.

6.19.3.10 windowSize()

```
std::uint64_t cs::LogInfo::windowSize ( ) const  [noexcept]
```

Read accessor for the window size.

Returns

The window size.

Definition at line 333 of file log_info.cpp.

6.19.4 Friends And Related Function Documentation

6.19.4.1 operator"!=

```
bool operator!= (
            const LogInfo & lhs,
            const LogInfo & rhs )  [friend]
```

Compares two LogInfos for inequality.

Parameters

lhs	The left hand side operand.
rhs	The right hand side operand.

Returns

true if `lhs` and `rhs` are considered not equal; otherwise false.

Definition at line 287 of file log_info.cpp.

6.19.4.2 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const LogInfo & logInfo )  [friend]
```

Prints a LogInfo to an ostream.

Parameters

os	The ostream to print to.
logInfo	The LogInfo to print.

Returns

os

Definition at line 292 of file log_info.cpp.

6.19.4.3 operator==

```
bool operator== (
            const LogInfo & lhs,
            const LogInfo & rhs )  [friend]
```

Compares two LogInfos for equality.

Parameters

lhs	The left hand side operand.
rhs	The right hand side operand.

Returns

true if `lhs` and `rhs` are considered equal; otherwise false.

Definition at line 264 of file log_info.cpp.

6.19.5 Member Data Documentation

6.19.5.1 invalidSensor

```
const std::uint64_t cs::LogInfo::invalidSensor = UINT64_C(0xFFFFFFFFFFFFFFFF)  [static]
```

Represents an invalid sensor.

Definition at line 25 of file log_info.hpp.

The documentation for this class was generated from the following files:

- compare_segmentation/include/log_info.hpp
- compare_segmentation/src/log_info.cpp

6.20 cs::LogLine Class Reference

A line out of a log file.

```
#include <log_line.hpp>
```

Public Member Functions

- std::uint64_t segmentationPointCount () const noexcept

  Read accessor for the segmentation point count.
- const cl::fs::Path & filePath () const noexcept

  Read accessor for the file path.
- cl::Expected< std::string > fileName () const

  Creates the short file name for the file in the log line.
- std::uint64_t sensor () const noexcept

  Read acccessor for the sensor.

Static Public Member Functions

- static cl::Expected< LogLine > parse (const std::string &line)

  Parses a LogLine out of a line of text read from a log file.

Static Public Attributes

- static const std::uint64_t invalidSensor = UINT64_C(0xFFFFFFFFFFFFFFFF)

  Indicates an invalid sensor.

6.20.1 Detailed Description

A line out of a log file.

Definition at line 14 of file log_line.hpp.

6.20.2 Member Function Documentation

6.20.2.1 fileName()

```
cl::Expected< std::string > cs::LogLine::fileName ( ) const
```

Creates the short file name for the file in the log line.

Returns

The resulting short file name or an error.

Definition at line 126 of file log_line.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.20.2.2 filePath()

```
const cl::fs::Path & cs::LogLine::filePath ( ) const  [noexcept]
```

Read accessor for the file path.

Returns

The file path of the file in the log line.

Definition at line 124 of file log_line.cpp.

Here is the caller graph for this function:

6.20.2.3 parse()

```
cl::Expected< LogLine > cs::LogLine::parse (
            const std::string & line )  [static]
```

Parses a LogLine out of a line of text read from a log file.

Parameters

line	The line read.

Returns

The resulting LogLine or an error.

Definition at line 31 of file log_line.cpp.

Here is the caller graph for this function:

6.20.2.4 segmentationPointCount()

```
std::uint64_t cs::LogLine::segmentationPointCount ( ) const  [noexcept]
```

Read accessor for the segmentation point count.

Returns

The segmentation point count.

Definition at line 119 of file log_line.cpp.

6.20.2.5 sensor()

```
std::uint64_t cs::LogLine::sensor ( ) const  [noexcept]
```

Read acccessor for the sensor.

Returns

The sensor.

Note

Will only return a valid sensor if the LogLine is for a preprocessed file.

Definition at line 164 of file log_line.cpp.

6.20.3 Member Data Documentation

6.20.3.1 invalidSensor

```
const std::uint64_t cs::LogLine::invalidSensor = UINT64_C(0xFFFFFFFFFFFFFFFF)  [static]
```

Indicates an invalid sensor.

Definition at line 19 of file log_line.hpp.

The documentation for this class was generated from the following files:

- compare_segmentation/include/log_line.hpp
- compare_segmentation/src/log_line.cpp

6.21 cm::ManualSegmentationPoint Class Reference

Type used to represent a manual segmentation point.

```
#include <manual_segmentation_point.hpp>
```

Public Member Functions

- ManualSegmentationPoint (std::uint32_t hour, std::uint32_t minute, std::uint32_t second, std::uint32_t frame)

  Creates a ManualSegmentationPoint.
- std::uint32_t hour () const noexcept

  Read accessor for the hour property.
- std::uint32_t minute () const noexcept

  Read accessor for the minute property.
- std::uint32_t second () const noexcept

  Read accessor for the second property.
- std::uint32_t frame () const noexcept

  Read accessor for the frame property.
- std::uint64_t asMilliseconds () const noexcept

  Converts this manual segmentation point into a millisecond representation.

Static Public Member Functions

- static std::unordered_map< DataSetIdentifier, std::vector< ManualSegmentationPoint > > readCsvFile ()

  Reads the CSV file of the manual segmentation points.
- static std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t > > convertToHardwareTimestamps (const std::unordered_map< DataSetIdentifier, std::vector< ManualSegmentationPoint >> &manual↩ SegmentationPoints, const std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >> &pythonResult)

  Converts `manualSegmentationPoints` imported from the CSV file to hardware timestamps.

Friends

- bool operator== (const ManualSegmentationPoint &lhs, const ManualSegmentationPoint &rhs) noexcept

  Compares two manual segmentation points for equality.
- bool operator!= (const ManualSegmentationPoint &lhs, const ManualSegmentationPoint &rhs) noexcept

  Compares two manual segmentation points for inequality.
- std::ostream & operator<< (std::ostream &os, const ManualSegmentationPoint &manualSegmentationPoint)

  Prints `manualSegmentationPoint` to `os`.

6.21.1 Detailed Description

Type used to represent a manual segmentation point.

Definition at line 17 of file manual_segmentation_point.hpp.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 ManualSegmentationPoint()

```
cm::ManualSegmentationPoint::ManualSegmentationPoint (
            std::uint32_t hour,
            std::uint32_t minute,
            std::uint32_t second,
            std::uint32_t frame )
```

Creates a ManualSegmentationPoint.

Parameters

hour	The hour to use. Must be within [0,59].
minute	The minute to use. Must be within [0,59].
second	The second to use. Must be within [0,59].
frame	The frame to use. Must be within [0,29].

Exceptions

cl::Exception	if one of the arguments is out of bounds.

Definition at line 417 of file manual_segmentation_point.cpp.

Here is the call graph for this function:

6.21.3 Member Function Documentation

6.21.3.1 asMilliseconds()

```
std::uint64_t cm::ManualSegmentationPoint::asMilliseconds ( ) const  [noexcept]
```

Converts this manual segmentation point into a millisecond representation.

Returns

This manual segmentation point converted to milliseconds.

Definition at line 475 of file manual_segmentation_point.cpp.

6.21.3.2 convertToHardwareTimestamps()

```
std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t > > cm::ManualSegmentation↩
Point::convertToHardwareTimestamps (
            const std::unordered_map< DataSetIdentifier, std::vector< ManualSegmentationPoint
>> & manualSegmentationPoints,
            const std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >> & python↩
Result )  [static]
```

Converts `manualSegmentationPoints` imported from the CSV file to hardware timestamps.

Parameters

manualSegmentationPoints	The manual segmentation points that were read from the CSV file.
pythonResult	The result from Python of a (good) `Configuration` to use for the first segmentation point in order to convert the manual segmentation points to ones that are based on hardware timestamps.

Returns

The resulting hardware timestamp based manual segementation points.

Exceptions

cl::Exception	on error.

Definition at line 361 of file manual_segmentation_point.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.21.3.3 frame()

```
std::uint32_t cm::ManualSegmentationPoint::frame ( ) const  [noexcept]
```

Read accessor for the frame property.

Returns

The frame within the second of this manual segmentation point.

Definition at line 470 of file manual_segmentation_point.cpp.

Here is the caller graph for this function:

6.21.3.4 hour()

```
std::uint32_t cm::ManualSegmentationPoint::hour ( ) const  [noexcept]
```

Read accessor for the hour property.

Returns

The hour.

Definition at line 458 of file manual_segmentation_point.cpp.

Here is the caller graph for this function:

6.21.3.5 minute()

```
std::uint32_t cm::ManualSegmentationPoint::minute ( ) const  [noexcept]
```

Read accessor for the minute property.

Returns

The minute.

Definition at line 460 of file manual_segmentation_point.cpp.

Here is the caller graph for this function:

6.21.3.6 readCsvFile()

```
std::unordered_map< DataSetIdentifier, std::vector< ManualSegmentationPoint > > cm::Manual↩
SegmentationPoint::readCsvFile ( )  [static]
```

Reads the CSV file of the manual segmentation points.

Returns

A map that maps the `DataSetIdentifier` enumerators to vectors of the corresponding manual segmentation points extracted from the CSV file.

Exceptions

cl::Exception	if parsing fails, CSV processing fails or the CSV file is missing.

Definition at line 211 of file manual_segmentation_point.cpp.

Here is the caller graph for this function:

6.21.3.7 second()

```
std::uint32_t cm::ManualSegmentationPoint::second ( ) const  [noexcept]
```

Read accessor for the second property.

Returns

The second.

Definition at line 465 of file manual_segmentation_point.cpp.

Here is the caller graph for this function:

6.21.4 Friends And Related Function Documentation

6.21.4.1 operator"!=

```
bool operator!= (
            const ManualSegmentationPoint & lhs,
            const ManualSegmentationPoint & rhs )  [friend]
```

Compares two manual segmentation points for inequality.

Parameters

lhs	The first operand.
rhs	The second operand.

Returns

true if `lhs` is considered not equal to `rhs`; false otherwise.

Definition at line 189 of file manual_segmentation_point.cpp.

6.21.4.2 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const ManualSegmentationPoint & manualSegmentationPoint )  [friend]
```

Prints `manualSegmentationPoint` to `os`.

Parameters

os	The ostream to print to
manualSegmentationPoint	The `ManualSegmentationPoint` to print.

Returns

`os`

Definition at line 196 of file manual_segmentation_point.cpp.

6.21.4.3 operator==

```
bool operator== (
            const ManualSegmentationPoint & lhs,
            const ManualSegmentationPoint & rhs )  [friend]
```

Compares two manual segmentation points for equality.

Parameters

lhs	The first operand.
rhs	The second operand.

Returns

true if `lhs` is considered equal to `rhs`; false otherwise.

Definition at line 181 of file manual_segmentation_point.cpp.

The documentation for this class was generated from the following files:

- confusion_matrix/include/manual_segmentation_point.hpp
- confusion_matrix/src/manual_segmentation_point.cpp

6.22 cl::fs::Path Class Reference

A filesystem path.

`#include <path.hpp>`

Public Member Functions

- Path ()

  Default constructs an (empty) path.
- PL_IMPLICIT Path (std::string path)

  Creates a path.
- PL_IMPLICIT Path (const char *path)

  Creates a path.
- bool exists () const noexcept

  Checks if the path exists.
- bool isFile () const noexcept

  Checks if the path is a file.
- bool isDirectory () const noexcept

  Checks if the path is a directory.
- const std::string & str () const noexcept

  Read accessor for the underlying string.

Friends

- std::ostream & operator<< (std::ostream &os, const Path &path)

  Prints a Path to an ostream.
- bool operator< (const Path &lhs, const Path &rhs) noexcept

  Checks if `lhs` is less than `rhs`.
- bool operator== (const Path &lhs, const Path &rhs) noexcept

  Equality compares `lhs` and `rhs`.

6.22.1 Detailed Description

A filesystem path.

Definition at line 14 of file path.hpp.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 Path() [1/3]

```
cl::fs::Path::Path ( )
```

Default constructs an (empty) path.

Definition at line 37 of file path.cpp.

6.22.2.2 Path() [2/3]

```
cl::fs::Path::Path (
            std::string path )
```

Creates a path.

Parameters

path	The string to construct from.

Definition at line 39 of file path.cpp.

6.22.2.3 Path() [3/3]

```
cl::fs::Path::Path (
            const char * path )
```

Creates a path.

Parameters

path	The string to construct from.

Definition at line 46 of file path.cpp.

6.22.3 Member Function Documentation

6.22.3.1 exists()

```
bool cl::fs::Path::exists ( ) const  [noexcept]
```

Checks if the path exists.

Returns

true if the path exists; otherwise false.

Definition at line 48 of file path.cpp.

Here is the call graph for this function:

6.22.3.2 isDirectory()

```
bool cl::fs::Path::isDirectory ( ) const  [noexcept]
```

Checks if the path is a directory.

Returns

true if the path is a directory; otherwise false.

Definition at line 104 of file path.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.22.3.3 isFile()

```
bool cl::fs::Path::isFile ( ) const  [noexcept]
```

Checks if the path is a file.

Returns

true if the path is a file; otherwise false.

Definition at line 77 of file path.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

6.22.3.4 str()

```
const std::string & cl::fs::Path::str ( ) const  [noexcept]
```

Read accessor for the underlying string.

Returns

The underlying string.

Definition at line 127 of file path.cpp.

Here is the caller graph for this function:

6.22.4 Friends And Related Function Documentation

6.22.4.1 operator<

```
bool operator< (
            const Path & lhs,
            const Path & rhs )  [friend]
```

Checks if `lhs` is less than `rhs`.

Parameters

lhs	The left hand side operand.
rhs	The right hand side operand.

Returns

true if lhs < rhs; otherwise false.

Definition at line 27 of file path.cpp.

6.22.4.2 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const Path & path )  [friend]
```

Prints a Path to an ostream.

Parameters

os	the ostream to print to.
path	The path to print.

Returns

`os`

Definition at line 22 of file path.cpp.

6.22.4.3 operator==

```
bool operator== (
            const Path & lhs,
            const Path & rhs )  [friend]
```

Equality compares `lhs` and `rhs`.

Parameters

lhs	The left hand side operand.
rhs	The right hand side operand.

Returns

true if `lhs` and `rhs` are equal.

Definition at line 32 of file path.cpp.

The documentation for this class was generated from the following files:

- csv_lib/include/cl/fs/path.hpp
- csv_lib/src/cl/fs/path.cpp

6.23 cl::Process Class Reference

```
#include <process.hpp>
```

Public Types

- using this_type = Process

Public Member Functions

- PL_NONCOPYABLE (Process)
- Process (this_type &&other) noexcept
- this_type & operator= (this_type &&other) noexcept
- ~Process ()
- std::FILE * file () noexcept
- const std::FILE * file () const noexcept

Static Public Member Functions

- static Expected< Process > create (pl::string_view command, pl::string_view mode)

6.23.1 Detailed Description

Definition at line 11 of file process.hpp.

6.23.2 Member Typedef Documentation

6.23.2.1 this_type

```
using cl::Process::this_type = Process
```

Definition at line 15 of file process.hpp.

6.23.3 Constructor & Destructor Documentation

6.23.3.1 Process()

```
cl::Process::Process (
            this_type && other )  [noexcept]
```

Definition at line 56 of file process.cpp.

6.23.3.2 ~Process()

```
cl::Process::~Process ( )
```

Definition at line 69 of file process.cpp.

6.23.4 Member Function Documentation

6.23.4.1 create()

```
Expected< Process > cl::Process::create (
            pl::string_view command,
            pl::string_view mode )  [static]
```

Definition at line 36 of file process.cpp.

Here is the caller graph for this function:

6.23.4.2 file() [1/2]

```
const std::FILE* cl::Process::file ( ) const  [noexcept]
```

6.23.4.3 file() [2/2]

```
const std::FILE * cl::Process::file ( )  [noexcept]
```

Definition at line 79 of file process.cpp.

6.23.4.4 operator=()

```
Process & cl::Process::operator= (
            this_type && other )  [noexcept]
```

Definition at line 61 of file process.cpp.

6.23.4.5 PL_NONCOPYABLE()

```
cl::Process::PL_NONCOPYABLE (
            Process  )
```

The documentation for this class was generated from the following files:

- csv_lib/include/cl/process.hpp
- csv_lib/src/cl/process.cpp

Chapter 7

File Documentation

7.1 compare_segmentation/CMakeLists.txt File Reference

Functions

- set (LIB_NAME compare_segmentation_lib) set(LIB_HEADERS include/csv_line.hpp include/data↩ _set_info.hpp include/filter_kind.hpp include/log_files.hpp include/log_info.hpp include/log_line.hpp include/mode.hpp include/paths.hpp include/segmentation_kind.hpp) set(LIB_SOURCES src/csv_line.↩ cpp src/data_set_info.cpp src/filter_kind.cpp src/log_files.cpp src/log_info.cpp src/log_line.cpp src/mode.cpp src/segmentation_kind.cpp) add_library($

7.1.1 Function Documentation

7.1.1.1 set()

```
set (
            LIB_NAME compare_segmentation_lib )
```

Definition at line 2 of file CMakeLists.txt.

7.2 compare_segmentation/test/CMakeLists.txt File Reference

Functions

- include (GoogleTest) set(TEST_NAME compare_segmentation_test) set(TEST_SOURCES csv_line_test.↩ cpp data_set_info_test.cpp log_files_test.cpp log_info_test.cpp log_line_test.cpp main.cpp mode_test.cpp) add_executable($

7.2.1 Function Documentation

7.2.1.1 include()

```
include (
            GoogleTest  )
```

Definition at line 1 of file CMakeLists.txt.

7.3 counting/CMakeLists.txt File Reference

Functions

- set (LIB_NAME counting_lib) set(LIB_HEADERS include/above_threshold.hpp include/average_↩ comparison_value_calculator.hpp include/half_maximum_comparison_value_calculator.hpp include/is_↩ relevant.hpp include/percentage_of.hpp include/run_above_threshold.hpp) set(LIB_SOURCES src/above↩ _threshold.cpp src/average_comparison_value_calculator.cpp src/half_maximum_comparison_value_↩ calculator.cpp src/run_above_threshold.cpp) add_library($

7.3.1 Function Documentation

7.3.1.1 set()

```
set (
            LIB_NAME counting_lib )
```

Definition at line 2 of file CMakeLists.txt.

7.4 counting/test/CMakeLists.txt File Reference

Functions

- include (GoogleTest) set(TEST_NAME counting_test) set(TEST_SOURCES above_threshold_test.cpp main.cpp percentage_of_test.cpp) add_executable($

7.4.1 Function Documentation

7.4.1.1 include()

```
include (
            GoogleTest  )
```

Definition at line 1 of file CMakeLists.txt.

7.5 csv_lib/CMakeLists.txt File Reference

Functions

- set (LIB_NAME csv_lib) set(LIB_HEADERS include/cl/fs/directory_listing.hpp include/cl/fs/file.hpp include/cl/fs/file_stream.hpp include/cl/fs/path.hpp include/cl/fs/separator.hpp include/cl/fs/windows.↩hpp include/cl/channel.hpp include/cl/column.hpp include/cl/data_point.hpp include/cl/data_set.hpp include/cl/dos2unix.hpp include/cl/error.hpp include/cl/exception.hpp include/cl/process.hpp include/cl/read↩_csv_file.hpp include/cl/s2n.hpp include/cl/sensor.hpp include/cl/to_string.hpp include/cl/use_unbuffered_↩io.hpp) set(LIB_SOURCES src/cl/fs/directory_listing.cpp src/cl/fs/file.cpp src/cl/fs/file_stream.cpp src/cl/fs/path.↩cpp src/cl/fs/windows.cpp src/cl/channel.cpp src/cl/data_point.cpp src/cl/data_set.cpp src/cl/dos2unix.cpp src/cl/error.cpp src/cl/exception.cpp src/cl/process.cpp src/cl/read_csv_file.cpp src/cl/sensor.cpp src/cl/use↩_unbuffered_io.cpp) add_library($

7.5.1 Function Documentation

7.5.1.1 set()

```
set (
            LIB_NAME csv_lib )
```

Definition at line 2 of file CMakeLists.txt.

7.6 csv_lib/test/CMakeLists.txt File Reference

Functions

- include (GoogleTest) set(TEST_NAME csv_lib_test) set(TEST_SOURCES channel_test.cpp column_test.↩cpp data_point_test.cpp directory_listing_test.cpp error_test.cpp exception_test.cpp main.cpp sensor_test.↩cpp to_string_test.cpp read_csv_file_test.cpp data_set_test.cpp s2n_test.cpp) add_executable($

7.6.1 Function Documentation

7.6.1.1 include()

```
include (
            GoogleTest  )
```

Definition at line 1 of file CMakeLists.txt.

7.7 fix_csv/CMakeLists.txt File Reference

Functions

- set (LIB_NAME fix_mogasens_csv_lib) set(LIB_HEADERS include/adjust_hardware_timestamp.↩ hpp include/convert_to_unix_line_endings.hpp include/create_backup_file.hpp include/delete_non↩ _bosch_sensors.hpp include/delete_out_of_bounds_values.hpp include/remove_zeros_from_field.hpp include/restore_from_backup.hpp include/write_file.hpp) set(LIB_SOURCES src/adjust_hardware_↩ timestamp.cpp src/convert_to_unix_line_endings.cpp src/create_backup_file.cpp src/delete_non_bosch↩ _sensors.cpp src/delete_out_of_bounds_values.cpp src/remove_zeros_from_field.cpp src/restore_from_↩ backup.cpp src/write_file.cpp) add_library($

7.7.1 Function Documentation

7.7.1.1 set()

```
set (
            LIB_NAME fix_mogasens_csv_lib )
```

Definition at line 2 of file CMakeLists.txt.

7.8 fix_csv/test/CMakeLists.txt File Reference

Functions

- include (GoogleTest) set(TEST_NAME fmc_test) set(TEST_SOURCES main.cpp remove_zeros_from_↩ field_test.cpp adjust_hardware_timestamp_test.cpp) add_executable($

7.8.1 Function Documentation

7.8.1.1 include()

```
include (
            GoogleTest  )
```

Definition at line 1 of file CMakeLists.txt.

7.9 confusion_matrix/CMakeLists.txt File Reference

Functions

- set (LIB_NAME confusion_matrix_lib) set(LIB_HEADERS include/closest_one.hpp include/configuration.hpp include/confusion_matrix.hpp include/confusion_matrix_best_configs.hpp include/create_segmentation_↩results.hpp include/csv_file_info.hpp include/data_set_identifier.hpp include/distance.hpp include/distance↩_score.hpp include/fetch.hpp include/imu.hpp include/interpolated_data_set_paths.hpp include/manual_↩segmentation_point.hpp include/order_configurations_by_quality.hpp include/python_output.hpp include/segment.↩hpp include/split_string.hpp) set(LIB_SOURCES src/closest_one.cpp src/configuration.cpp src/confusion_↩matrix.cpp src/confusion_matrix_best_configs.cpp src/create_segmentation_results.cpp src/csv_file_info.cpp src/data_set_identifier.cpp src/distance.cpp src/distance_score.cpp src/imu.cpp src/interpolated_data_↩set_paths.cpp src/manual_segmentation_point.cpp src/order_configurations_by_quality.cpp src/python_↩output.cpp src/segment.cpp src/split_string.cpp) add_library($

7.9.1 Function Documentation

7.9.1.1 set()

```
set (
            LIB_NAME confusion_matrix_lib )
```

Definition at line 2 of file CMakeLists.txt.

7.10 confusion_matrix/test/CMakeLists.txt File Reference

Functions

- include (GoogleTest) set(TEST_NAME confusion_matrix_test) set(TEST_SOURCES data_set_identifier_↩test.cpp interpolated_data_set_paths_test.cpp main.cpp manual_segmentation_point_test.cpp segment_↩test.cpp split_string_test.cpp) add_executable($

7.10.1 Function Documentation

7.10.1.1 include()

```
include (
            GoogleTest  )
```

Definition at line 1 of file CMakeLists.txt.

7.11 compare_segmentation/include/csv_line.hpp File Reference

```
#include <cstdint>
#include <string>
#include <vector>
#include <tl/optional.hpp>
#include "filter_kind.hpp"
#include "segmentation_kind.hpp"
```

Include dependency graph for csv_line.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- class cs::CsvLineBuilder

  Builder for a CSV line.

Namespaces

- cs

7.12 compare_segmentation/include/data_set_info.hpp File Reference

```
#include <cstdint>
#include <stdexcept>
#include <pl/except.hpp>
#include <pl/string_view.hpp>
```

Include dependency graph for data_set_info.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- struct cs::data_set_info< Tag >

  Meta function for data set tags.

Namespaces

- cs

Macros

- #define CS_SPECIALIZE_DATA_SET_INFO(tag, string, repetitionCount)

Functions

- cs::PL_DEFINE_EXCEPTION_TYPE (NoSuchDataSetException, std::logic_error)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Felix1, "11.17.39", 24)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Felix2, "12.50.00", 20)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Felix3, "13.00.09", 15)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Marcelle1, "14.59.59", 10)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Marcelle2, "15.13.22", 16)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Marcelle3, "15.31.36", 18)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Mike1, "14.07.33", 26)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Mike2, "14.14.32", 22)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Mike3, "14.20.28", 18)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Andre1, "Andre_liegestuetzen1", 27)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Andre2, "Andre_liegestuetzen2", 20)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Andre3, "Andre_liegestuetzen3", 17)
- cs::CS_SPECIALIZE_DATA_SET_INFO (AndreSquats1, "Andre_Squats", 30)
- cs::CS_SPECIALIZE_DATA_SET_INFO (AndreSquats2, "Andre_Squats2", 49)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Jan1, "Jan_liegestuetzen1", 25)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Jan2, "Jan_liegestuetzen2", 19)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Jan3, "Jan_liegestuetzen3", 13)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Lucas1, "Lukas_liegestuetzen1", 24)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Lucas2, "Lukas_liegestuetzen2", 19)
- cs::CS_SPECIALIZE_DATA_SET_INFO (Lucas3, "Lukas_liegestuetzen3", 11)
- std::uint64_t cs::repetitionCount (pl::string_view dataSet)

  Fetches the repetition count for a given data set identified by its string.

7.12.1 Macro Definition Documentation

7.12.1.1 CS_SPECIALIZE_DATA_SET_INFO

```
#define CS_SPECIALIZE_DATA_SET_INFO(
                tag,
                string,
                repetitionCount )
```

Value:

```
  struct tag {                                                                  \
  };                                                                            \
  constexpr bool contains##tag(pl::string_view other)                           \
  {                                                                             \
    return other.contains(string);                                              \
  }                                                                             \
  template<>                                                                    \
  struct data_set_info<tag> {                                                   \
    static constexpr pl::string_view text        = string;                      \
    static constexpr std::uint64_t   repetitions = UINT64_C(repetitionCount); \
  }
```

Definition at line 23 of file data_set_info.hpp.

7.13 compare_segmentation/include/filter_kind.hpp File Reference

```
#include <iosfwd>
```

Include dependency graph for filter_kind.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cs

Enumerations

- enum cs::FilterKind { cs::FilterKind::Butterworth, cs::FilterKind::MovingAverage }

  Type for the different kinds of filters.

Functions

- std::ostream & cs::operator<< (std::ostream &os, FilterKind filterKind)

  Prints a FilterKind to an ostream.

7.14 compare_segmentation/include/log_files.hpp File Reference

```
#include <vector>
#include <pl/string_view.hpp>
#include <cl/error.hpp>
#include <cl/fs/path.hpp>
```

Include dependency graph for log_files.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cs

Functions

- cl::Expected< std::vector< cl::fs::Path > > cs::logFiles (pl::string_view directoryPath)

  Fetches the paths to the log files in the given directory.

7.15 compare_segmentation/include/log_info.hpp File Reference

```
#include <cstdint>
#include <iosfwd>
#include <cl/error.hpp>
#include <cl/fs/path.hpp>
#include "filter_kind.hpp"
```

```
#include "segmentation_kind.hpp"
```

Include dependency graph for log_info.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- class cs::LogInfo

  Information about a log file.

Namespaces

- cs

7.16 compare_segmentation/include/log_line.hpp File Reference

```
#include <cstdint>
#include <string>
#include "cl/error.hpp"
#include "cl/fs/path.hpp"
```

Include dependency graph for log_line.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- class cs::LogLine

  A line out of a log file.

Namespaces

- cs

7.17 compare_segmentation/include/mode.hpp File Reference

```
#include <iosfwd>
#include <cl/error.hpp>
```

Include dependency graph for mode.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cs

Macros

- #define CS_MODE
- #define CS_MODE_X(enm) enm,

Enumerations

- enum cs::Mode { cs::Mode::CS_MODE_X, cs::Mode::CS_MODE }

  Enumerator type for the different modes of the compare_segmentation application.

Functions

- std::ostream & cs::operator<< (std::ostream &os, Mode mode)

  Prints a `Mode` to an ostream.
- cl::Expected< Mode > cs::parseMode (const char *szCmdArg)

  Parses a null-terminated byte character string as a `Mode`.

7.17.1 Macro Definition Documentation

7.17.1.1 CS_MODE

```
#define CS_MODE
```

Value:

```
CS_MODE_X(AllDataSets)  \
CS_MODE_X(AllPushUps)   \
CS_MODE_X(PushUps250Hz) \
CS_MODE_X(PushUps200Hz) \
CS_MODE_X(Squats)
```

Definition at line 8 of file mode.hpp.

7.17.1.2 CS_MODE_X

```
#define CS_MODE_X(
            enm ) enm,
```

Definition at line 20 of file mode.hpp.

7.18 compare_segmentation/include/paths.hpp File Reference

```
#include <pl/string_view.hpp>
```

Include dependency graph for paths.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cs

Variables

- constexpr pl::string_view cs::logPath {"segmentation_comparison/logs"}

  Relative path to the directory containing the preprocessed log files.
- constexpr pl::string_view cs::oldLogPath {"segmentation_comparison/logs/old"}

  Relative path to the directory containing the old log files.

7.19 compare_segmentation/include/segmentation_kind.hpp File Reference

```
#include <iosfwd>
#include <pl/bitmask.hpp>
#include <pl/byte.hpp>
```

Include dependency graph for segmentation_kind.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cs

Enumerations

- enum cs::SegmentationKind : pl::byte { cs::SegmentationKind::Minima = 0b0000'0001, cs::SegmentationKind::Maxima = 0b0000'0010, cs::SegmentationKind::Both = Minima | Maxima }

  The segmentation kind.

Functions

- std::ostream & cs::operator<< (std::ostream &os, SegmentationKind segmentationKind)

  Prints a SegmentationKind to an ostream.

7.20 compare_segmentation/src/csv_line.cpp File Reference

```
#include <cassert>
#include <cstddef>
#include <string>
#include <utility>
#include <tl/casts.hpp>
#include <pl/algo/ranged_algorithms.hpp>
#include "cl/sensor.hpp"
#include "cl/to_string.hpp"
#include "csv_line.hpp"
```

Include dependency graph for csv_line.cpp:

Namespaces

- cs

7.21 compare_segmentation/src/data_set_info.cpp File Reference

```
#include <cassert>
#include <cstdio>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include "data_set_info.hpp"
```

Include dependency graph for data_set_info.cpp:

Namespaces

- cs

Functions

- std::uint64_t cs::repetitionCount (pl::string_view dataSet)

  Fetches the repetition count for a given data set identified by its string.

7.22 compare_segmentation/src/filter_kind.cpp File Reference

```
#include <ostream>
#include "filter_kind.hpp"
```

Include dependency graph for filter_kind.cpp:

Namespaces

- cs

Functions

- std::ostream & cs::operator<< (std::ostream &os, FilterKind filterKind)

  Prints a FilterKind to an ostream.

7.23 compare_segmentation/src/log_files.cpp File Reference

```
#include <pl/algo/erase.hpp>
#include <pl/algo/ranged_algorithms.hpp>
#include <cl/fs/directory_listing.hpp>
```

```
#include "log_files.hpp"
```

Include dependency graph for log_files.cpp:

Namespaces

- cs

Functions

- cl::Expected< std::vector< cl::fs::Path > > cs::logFiles (pl::string_view directoryPath)

  Fetches the paths to the log files in the given directory.

7.24 compare_segmentation/src/log_info.cpp File Reference

```
#include <cstddef>
#include <cstring>
#include <ostream>
#include <regex>
#include <tuple>
#include <utility>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/size_t.hpp>
#include <pl/strcontains.hpp>
#include <pl/string_view.hpp>
#include "cl/fs/separator.hpp"
#include "cl/s2n.hpp"
#include "log_info.hpp"
#include "paths.hpp"
```

Include dependency graph for log_info.cpp:

Namespaces

- cs

Functions

- bool cs::operator== (const LogInfo &lhs, const LogInfo &rhs) noexcept
- bool cs::operator!= (const LogInfo &lhs, const LogInfo &rhs) noexcept
- std::ostream & cs::operator<< (std::ostream &os, const LogInfo &logInfo)

7.25 compare_segmentation/src/log_line.cpp File Reference

```
#include <cstddef>
#include <regex>
#include <string>
#include <utility>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/strcontains.hpp>
#include <pl/string_view.hpp>
#include "cl/s2n.hpp"
#include "log_line.hpp"
```

Include dependency graph for log_line.cpp:

Namespaces

- cs

7.26 compare_segmentation/src/main.cpp File Reference

```
#include <cmath>
#include <cstdint>
#include <cstdio>
#include <cstdlib>
#include <fstream>
#include <string>
#include <vector>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <csv.hpp>
```

```
#include <pl/algo/ranged_algorithms.hpp>
#include "cl/fs/separator.hpp"
#include "cl/to_string.hpp"
#include "csv_line.hpp"
#include "data_set_info.hpp"
#include "log_files.hpp"
#include "log_info.hpp"
#include "log_line.hpp"
#include "mode.hpp"
#include "paths.hpp"
```

Include dependency graph for main.cpp:

Functions

- int main (int argc, char *argv[])

7.26.1 Function Documentation

7.26.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Definition at line 28 of file main.cpp.

Here is the call graph for this function:

7.27 compare_segmentation/test/main.cpp File Reference

```
#include "gtest/gtest.h"
```

Include dependency graph for main.cpp:

Functions

- int main (int argc, char *argv[ ])

7.27.1 Function Documentation

7.27.1.1 main()

```
int main (
            int argc,
            char * argv[ ] )
```

Definition at line 3 of file main.cpp.

7.28 counting/src/main.cpp File Reference

```
#include <cassert>
#include <cstdio>
#include <cstdlib>
#include <array>
#include <fstream>
#include <sstream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/string_view.hpp>
#include "cl/channel.hpp"
#include "cl/data_set.hpp"
#include "cl/read_csv_file.hpp"
#include "cl/sensor.hpp"
#include "average_comparison_value_calculator.hpp"
#include "half_maximum_comparison_value_calculator.hpp"
```

```
#include "is_relevant.hpp"
#include "run_above_threshold.hpp"
```

Include dependency graph for main.cpp:

Functions

- int main (int argc, char ∗argv[ ])

7.28.1 Function Documentation

7.28.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Definition at line 24 of file main.cpp.

Here is the call graph for this function:

7.29 counting/test/main.cpp File Reference

```
#include "gtest/gtest.h"
```

Include dependency graph for main.cpp:

Functions

- int main (int argc, char ∗argv[ ])

7.29.1 Function Documentation

7.29.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Definition at line 3 of file main.cpp.

7.30 csv_lib/test/main.cpp File Reference

`#include "gtest/gtest.h"`
Include dependency graph for main.cpp:

Functions

- int main (int argc, char *argv[ ])

7.30.1 Function Documentation

7.30.1.1 main()

```
int main (
            int argc,
            char * argv[ ] )
```

Definition at line 3 of file main.cpp.

7.31 fix_csv/src/main.cpp File Reference

```
#include <cstdio>
#include <cstdlib>
#include <fstream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/algo/clamp.hpp>
#include <pl/numeric.hpp>
#include <pl/string_view.hpp>
#include "cl/column.hpp"
#include "cl/read_csv_file.hpp"
#include "cl/use_unbuffered_io.hpp"
#include "adjust_hardware_timestamp.hpp"
#include "convert_to_unix_line_endings.hpp"
#include "create_backup_file.hpp"
#include "delete_non_bosch_sensors.hpp"
#include "delete_out_of_bounds_values.hpp"
#include "remove_zeros_from_field.hpp"
#include "restore_from_backup.hpp"
#include "write_file.hpp"
```

Include dependency graph for main.cpp:

Functions

- int main (int argc, char *argv[ ])

7.31.1 Function Documentation

7.31.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Definition at line 26 of file main.cpp.

Here is the call graph for this function:

7.32 fix_csv/test/main.cpp File Reference

```
#include "gtest/gtest.h"
```

Include dependency graph for main.cpp:

Functions

- int main (int argc, char *argv[ ])

7.32.1 Function Documentation

7.32.1.1 main()

```
int main (
            int argc,
            char * argv[ ] )
```

Definition at line 3 of file main.cpp.

7.33 confusion_matrix/src/main.cpp File Reference

```
#include <cstdlib>
#include <fstream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/string_view.hpp>
#include <pl/unused.hpp>
#include <cl/fs/file.hpp>
#include <cl/use_unbuffered_io.hpp>
#include "confusion_matrix_best_configs.hpp"
#include "create_segmentation_results.hpp"
#include "csv_file_info.hpp"
#include "fetch.hpp"
#include "interpolated_data_set_paths.hpp"
#include "manual_segmentation_point.hpp"
#include "order_configurations_by_quality.hpp"
```

Include dependency graph for main.cpp:

Macros

- #define SORT_PRINT(kind)

Functions

- int main (int argc, char *argv[ ])

  Entry point.

7.33.1 Macro Definition Documentation

7.33.1.1 SORT_PRINT

```
#define SORT_PRINT(
            kind )
```

Value:

```
pl::algo::stable_sort(bestConfigs, cm::kind##Sorter);                \
print("{}\n", #kind);                                                \
for (const cm::ConfigWithTotalConfusionMatrix& cur : bestConfigs) { \
  print("{}\n", cur);                                                \
}                                                                    \
print("\nBest configuration (" #kind "): {}\n", bestConfigs.front())
```

7.33.2 Function Documentation

7.33.2.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Entry point.

Parameters

argc	Argument count.
argv	Argument values.

Returns

EXIT_SUCCESS on success; otherwise EXIT_FAILURE.

Definition at line 54 of file main.cpp.

Here is the call graph for this function:

7.34 confusion_matrix/test/main.cpp File Reference

```
#include "gtest/gtest.h"
```

Include dependency graph for main.cpp:

Functions

- int main (int argc, char ∗argv[ ])

7.34.1 Function Documentation

7.34.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Definition at line 3 of file main.cpp.

7.35 compare_segmentation/src/mode.cpp File Reference

```
#include <cstring>
#include <ostream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/unreachable.hpp>
#include "mode.hpp"
```

Include dependency graph for mode.cpp:

Namespaces

- cs

Macros

- #define CS_MODE_X(enm) case Mode::enm: return os << #enm;
- #define CS_MODE_X(enm) if (std::strcmp(#enm, szCmdArg) == 0) { return Mode::enm; }

Functions

- std::ostream & cs::operator<< (std::ostream &os, Mode mode)

  Prints a `Mode` to an ostream.
- cl::Expected< Mode > cs::parseMode (const char *szCmdArg)

  Parses a null-terminated byte character string as a `Mode`.

7.35.1 Macro Definition Documentation

7.35.1.1 CS_MODE_X [1/2]

```
#define CS_MODE_X(
            enm ) case Mode::enm:  return os << #enm;
```

7.35.1.2 CS_MODE_X [2/2]

```
#define CS_MODE_X(
            enm ) if (std::strcmp(#enm, szCmdArg) == 0) { return Mode::enm; }
```

7.36 compare_segmentation/src/segmentation_kind.cpp File Reference

```
#include <ostream>
#include "segmentation_kind.hpp"
```

Include dependency graph for segmentation_kind.cpp:

Namespaces

- cs

Functions

- std::ostream & cs::operator<< (std::ostream &os, SegmentationKind segmentationKind)

  Prints a SegmentationKind to an ostream.

7.37 compare_segmentation/test/csv_line_test.cpp File Reference

```
#include <array>
#include "gtest/gtest.h"
#include <pl/unreachable.hpp>
#include "csv_line.hpp"
```

Include dependency graph for csv_line_test.cpp:

Functions

- TEST (CsvLine, shouldWork)

7.37.1 Function Documentation

7.37.1.1 TEST()

```
TEST (
            CsvLine ,
            shouldWork  )
```

Definition at line 30 of file csv_line_test.cpp.

7.38 compare_segmentation/test/data_set_info_test.cpp File Reference

```
#include "gtest/gtest.h"
#include "data_set_info.hpp"
```

Include dependency graph for data_set_info_test.cpp:

Functions

- TEST (dataSetInfo, repetitionCount)

7.38.1 Function Documentation

7.38.1.1 TEST()

```
TEST (
            dataSetInfo ,
            repetitionCount  )
```

Definition at line 5 of file data_set_info_test.cpp.

Here is the call graph for this function:

7.39 compare_segmentation/test/log_files_test.cpp File Reference

```
#include "gtest/gtest.h"
#include <log_files.hpp>
#include <paths.hpp>
```

Include dependency graph for log_files_test.cpp:

Functions

- TEST (logFiles, shouldFindLogFiles)
- TEST (logFiles, shouldFindOldLogFiles)
- TEST (logFiles, shouldNotFindGarbage)

7.39.1 Function Documentation

7.39.1.1 TEST() [1/3]

```
TEST (
            logFiles ,
            shouldFindLogFiles  )
```

Definition at line 6 of file log_files_test.cpp.

Here is the call graph for this function:

7.39.1.2 TEST() [2/3]

```
TEST (
            logFiles ,
            shouldFindOldLogFiles  )
```

Definition at line 23 of file log_files_test.cpp.

Here is the call graph for this function:

7.39.1.3 TEST() [3/3]

```
TEST (
            logFiles ,
            shouldNotFindGarbage  )
```

Definition at line 46 of file log_files_test.cpp.

Here is the call graph for this function:

7.40 compare_segmentation/test/log_info_test.cpp File Reference

```
#include "gtest/gtest.h"
#include <pl/size_t.hpp>
#include <pl/strcontains.hpp>
#include "log_info.hpp"
```

Include dependency graph for log_info_test.cpp:

Functions

- TEST (LogInfo, shouldWork)
- TEST (LogInfo, shouldWork2)
- TEST (LogInfo, shouldWork3)
- TEST (LogInfo, shouldWork4)
- TEST (LogInfo, shouldWork5)
- TEST (LogInfo, shouldWork6)
- TEST (LogInfo, shouldWork7)
- TEST (LogInfo, shouldWork8)
- TEST (LogInfo, shouldWork9)
- TEST (LogInfo, shouldWorkWithOldPath)
- TEST (LogInfo, shouldWorkWithOldPath2)
- TEST (LogInfo, shouldResultInErrorIfLogFilePathIsTooShort)

- TEST (LogInfo, shouldFailIfSkipWindowIsInvalid)
- TEST (LogInfo, shouldFailIfDeleteTooCloseIsInvalid)
- TEST (LogInfo, shouldFailIfDeleteTooLowVarianceIsInvalid)
- TEST (LogInfo, shouldFailIfSegmentationKindIsInvalid)
- TEST (LogInfo, shouldFailIfWindowSizeIsInvalid)
- TEST (LogInfo, shouldFailIfFilterIsInvalid)
- TEST (LogInfo, shouldCreateUninitialzedObjectWhenDefaultConstructorIsCalled)

7.40.1 Function Documentation

7.40.1.1 TEST() [1/19]

```
TEST (
            LogInfo ,
            shouldCreateUninitialzedObjectWhenDefaultConstructorIsCalled  )
```

Definition at line 388 of file log_info_test.cpp.

7.40.1.2 TEST() [2/19]

```
TEST (
            LogInfo ,
            shouldFailIfDeleteTooCloseIsInvalid  )
```

Definition at line 341 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.3 TEST() [3/19]

```
TEST (
            LogInfo ,
            shouldFailIfDeleteTooLowVarianceIsInvalid  )
```

Definition at line 350 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.4 TEST() [4/19]

```
TEST (
            LogInfo ,
            shouldFailIfFilterIsInvalid  )
```

Definition at line 379 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.5 TEST() [5/19]

```
TEST (
            LogInfo ,
            shouldFailIfSegmentationKindIsInvalid  )
```

Definition at line 359 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.6 TEST() [6/19]

```
TEST (
            LogInfo ,
            shouldFailIfSkipWindowIsInvalid  )
```

Definition at line 332 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.7 TEST() [7/19]

```
TEST (
            LogInfo ,
            shouldFailIfWindowSizeIsInvalid  )
```

Definition at line 368 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.8 TEST() [8/19]

```
TEST (
            LogInfo ,
            shouldResultInErrorIfLogFilePathIsTooShort  )
```

Definition at line 325 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.9 TEST() [9/19]

```
TEST (
            LogInfo ,
            shouldWork  )
```

Definition at line 8 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.10 TEST() [10/19]

```
TEST (
            LogInfo ,
            shouldWork2  )
```

Definition at line 37 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.11 TEST() [11/19]

```
TEST (
            LogInfo ,
            shouldWork3  )
```

Definition at line 66 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.12 TEST() [12/19]

```
TEST (
            LogInfo ,
            shouldWork4  )
```

Definition at line 95 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.13 TEST() [13/19]

```
TEST (
            LogInfo ,
            shouldWork5  )
```

Definition at line 124 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.14 TEST() [14/19]

```
TEST (
            LogInfo ,
            shouldWork6  )
```

Definition at line 153 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.15 TEST() [15/19]

```
TEST (
            LogInfo ,
            shouldWork7  )
```

Definition at line 182 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.16 TEST() [16/19]

```
TEST (
            LogInfo ,
            shouldWork8  )
```

Definition at line 211 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.17 TEST() [17/19]

```
TEST (
            LogInfo ,
            shouldWork9  )
```

Definition at line 240 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.18 TEST() [18/19]

```
TEST (
            LogInfo ,
            shouldWorkWithOldPath  )
```

Definition at line 269 of file log_info_test.cpp.

Here is the call graph for this function:

7.40.1.19 TEST() [19/19]

```
TEST (
            LogInfo ,
            shouldWorkWithOldPath2  )
```

Definition at line 297 of file log_info_test.cpp.

Here is the call graph for this function:

7.41 compare_segmentation/test/log_line_test.cpp File Reference

```
#include <string>
#include "gtest/gtest.h"
#include "log_line.hpp"
```

Include dependency graph for log_line_test.cpp:

Functions

- TEST (LogLine, shouldWorkWithPreprocessedLine)
- TEST (LogLine, shouldWorkWithOldLine)
- TEST (LogLine, shouldNotMatchGarbage)
- TEST (LogLine, shouldNotParseGarbageSensor)

7.41.1 Function Documentation

7.41.1.1 TEST() [1/4]

```
TEST (
            LogLine ,
            shouldNotMatchGarbage  )
```

Definition at line 41 of file log_line_test.cpp.

Here is the call graph for this function:

7.41.1.2 TEST() [2/4]

```
TEST (
            LogLine ,
            shouldNotParseGarbageSensor  )
```

Definition at line 48 of file log_line_test.cpp.

Here is the call graph for this function:

7.41.1.3 TEST() [3/4]

```
TEST (
            LogLine ,
            shouldWorkWithOldLine  )
```

Definition at line 25 of file log_line_test.cpp.

Here is the call graph for this function:

7.41.1.4 TEST() [4/4]

```
TEST (
            LogLine ,
            shouldWorkWithPreprocessedLine  )
```

Definition at line 9 of file log_line_test.cpp.

Here is the call graph for this function:

7.42 compare_segmentation/test/mode_test.cpp File Reference

```
#include "gtest/gtest.h"
#include <cl/to_string.hpp>
```

```
#include "mode.hpp"
```

Include dependency graph for mode_test.cpp:

Functions

- TEST (Mode, shouldPrintCorrectly)
- TEST (Mode, shouldParseCorrectly)

7.42.1 Function Documentation

7.42.1.1 TEST() [1/2]

```
TEST (
            Mode ,
            shouldParseCorrectly  )
```

Definition at line 18 of file mode_test.cpp.

Here is the call graph for this function:

7.42.1.2 TEST() [2/2]

```
TEST (
            Mode ,
            shouldPrintCorrectly  )
```

Definition at line 7 of file mode_test.cpp.

Here is the call graph for this function:

7.43 confusion_matrix/include/closest_one.hpp File Reference

```
#include <cstdint>
#include <vector>
```

Include dependency graph for closest_one.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cm

Functions

- std::uint64_t cm::closestOne (std::uint64_t algorithmicallyDeterminedSegmentationPoint, const std::vector< std::uint64_t > &manualSegmentationPoints)

  Finds the segmentation point in `manualSegmentationPoints` *that is the closest to* `algorithmically↩ DeterminedSegmentationPoint`.

7.44 confusion_matrix/include/configuration.hpp File Reference

```
#include <cstddef>
#include <deque>
#include <iosfwd>
#include <string>
#include <vector>
#include <tl/optional.hpp>
#include <pl/hash.hpp>
#include <pl/total_order.hpp>
#include <cl/fs/path.hpp>
#include "imu.hpp"
```

Include dependency graph for configuration.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- class cm::Configuration

  Represents a possible configuration for the Python segmentor.
- class cm::Configuration::Builder

  Builder type for `Configuration`.
- struct std::hash<::cm::Configuration >

Namespaces

- cm

7.45 confusion_matrix/include/confusion_matrix.hpp File Reference

`#include <cstdint>`

Include dependency graph for confusion_matrix.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- class cm::ConfusionMatrix

  Type to represent a confusion matrix.

Namespaces

- cm

7.46 confusion_matrix/include/confusion_matrix_best_configs.hpp File Reference

```
#include <functional>
#include <iosfwd>
#include <unordered_map>
#include <vector>
#include "configuration.hpp"
#include "confusion_matrix.hpp"
#include "data_set_identifier.hpp"
```

Include dependency graph for confusion_matrix_best_configs.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- struct cm::ConfigWithTotalConfusionMatrix

  A `Configuration` with a `ConfusionMatrix`.

Namespaces

- cm

Macros

- #define CM_SORTER(criterion, op)

  Macro to define a sorter based on a single criterion for `ConfigWithTotalConfusionMatrix` objects.

Functions

- cm::CM_SORTER (truePositives, >)

  Sorts `ConfigWithTotalConfusionMatrix` objects by true positives (highest first)
- cm::CM_SORTER (trueNegatives, >)

  Sorts `ConfigWithTotalConfusionMatrix` objects by true negatives (highest first)
- cm::CM_SORTER (falsePositives,<)

  Sorts `ConfigWithTotalConfusionMatrix` objects by false positives (lowest first)
- cm::CM_SORTER (falseNegatives,<)

  Sorts `ConfigWithTotalConfusionMatrix` objects by false negatives (lowest first)
- std::vector< ConfigWithTotalConfusionMatrix > cm::confusionMatrixBestConfigs (const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t >> &manualSegmentationPoints, const std::unordered_map< Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >>> &algorithmically DeterminedSegmentationPoints, const std::function< bool(const ConfigWithTotalConfusionMatrix &, const ConfigWithTotalConfusionMatrix &)> &sorter)

  Determines the 'best' configurations.

Variables

- struct {
  } cm::disregardTrueNegativesSorter

  Sorter to sort `ConfigWithTotalConfusionMatrix` objects by the count of true positives minus the count of false positives minus the count of false negatives.
- struct {
  } cm::addTrueSubtractFalseSorter

  Sorter to sort `ConfigWithTotalConfusionMatrix` objects by the sum of true positives and true negatives minus the false positives minus the false negatives.

7.46.1 Macro Definition Documentation

7.46.1.1 CM_SORTER

```
#define CM_SORTER(
            criterion,
            op )
```

Value:

```
  inline constexpr struct {                                          \
    [[nodiscard]] bool operator()(                                   \
      const ConfigWithTotalConfusionMatrix& lhs,                     \
      const ConfigWithTotalConfusionMatrix& rhs) const noexcept      \
    {                                                                \
      if (                                                           \
        !(lhs.matrix.criterion() op rhs.matrix.criterion())          \
        && !(rhs.matrix.criterion() op lhs.matrix.criterion())) {    \
        return lhs.config < rhs.config;                              \
      }                                                              \
                                                                     \
      return lhs.matrix.criterion() op rhs.matrix.criterion();       \
    }                                                                \
  } criterion##Sorter
```

Macro to define a sorter based on a single criterion for `ConfigWithTotalConfusionMatrix` objects.

Definition at line 50 of file confusion_matrix_best_configs.hpp.

7.47 confusion_matrix/include/create_segmentation_results.hpp File Reference

```
#include <cstdint>
#include <unordered_map>
#include <vector>
#include <cl/fs/path.hpp>
```

```
#include "configuration.hpp"
```

Include dependency graph for create_segmentation_results.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cm

Functions

- std::unordered_map< cm::Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > > cm::createSegmentationResults ()

  Invokes Python to generate the segmentation points algorithmically.

7.48 confusion_matrix/include/csv_file_info.hpp File Reference

```
#include <cstdint>
#include <vector>
```

```
#include <cl/fs/path.hpp>
```

Include dependency graph for csv_file_info.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- class cm::CsvFileInfo

  Type to hold the hardware timestamps of a CSV file.

Namespaces

- cm

7.49 confusion_matrix/include/data_set_identifier.hpp File Reference

```
#include <iosfwd>
#include <cl/fs/path.hpp>
```

Include dependency graph for data_set_identifier.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cm

Macros

- #define CM_DATA_SET_IDENTIFIER
- #define CM_DATA_SET_IDENTIFIER_X(enm) enm,

Enumerations

- enum cm::DataSetIdentifier { cm::DataSetIdentifier::CM_DATA_SET_IDENTIFIER_X, cm::DataSetIdentifier::CM_DATA_SET_I
  }

  Scoped enum type for the data set identifiers.

Functions

- std::ostream & cm::operator<< (std::ostream &os, DataSetIdentifier dsi)

  Prints a DataSetIdentifier to an ostream.
- DataSetIdentifier cm::toDataSetIdentifier (const cl::fs::Path &path)

  Converts a path to a CSV file to the corresponding DataSetIdentifier.

7.49.1 Macro Definition Documentation

7.49.1.1 CM_DATA_SET_IDENTIFIER

```
#define CM_DATA_SET_IDENTIFIER
```

Value:

```
CM_DATA_SET_IDENTIFIER_X(Felix_11_17_39) \
CM_DATA_SET_IDENTIFIER_X(Felix_12_50_00) \
CM_DATA_SET_IDENTIFIER_X(Felix_13_00_09) \
CM_DATA_SET_IDENTIFIER_X(Mike_14_07_33)  \
CM_DATA_SET_IDENTIFIER_X(Mike_14_14_32)  \
CM_DATA_SET_IDENTIFIER_X(Mike_14_20_28)  \
CM_DATA_SET_IDENTIFIER_X(Marsi_14_59_59) \
CM_DATA_SET_IDENTIFIER_X(Marsi_15_13_22) \
CM_DATA_SET_IDENTIFIER_X(Marsi_15_31_36) \
CM_DATA_SET_IDENTIFIER_X(Jan_1)          \
CM_DATA_SET_IDENTIFIER_X(Jan_2)          \
CM_DATA_SET_IDENTIFIER_X(Jan_3)          \
CM_DATA_SET_IDENTIFIER_X(Andre_1)        \
CM_DATA_SET_IDENTIFIER_X(Andre_2)        \
CM_DATA_SET_IDENTIFIER_X(Andre_3)        \
CM_DATA_SET_IDENTIFIER_X(Andre_Squats_1) \
CM_DATA_SET_IDENTIFIER_X(Andre_Squats_2) \
CM_DATA_SET_IDENTIFIER_X(Lucas_1)        \
CM_DATA_SET_IDENTIFIER_X(Lucas_2)        \
CM_DATA_SET_IDENTIFIER_X(Lucas_3)
```

Definition at line 8 of file data_set_identifier.hpp.

7.49.1.2 CM_DATA_SET_IDENTIFIER_X

```
#define CM_DATA_SET_IDENTIFIER_X(
              enm ) enm,
```

Definition at line 34 of file data_set_identifier.hpp.

7.50 confusion_matrix/include/distance.hpp File Reference

```
#include <cstdint>
```

Include dependency graph for distance.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cm

Functions

- std::uint64_t cm::distance (std::uint64_t a, std::uint64_t b)

  Calculates the distance between `a` *and* `b`*.*

7.51 confusion_matrix/include/distance_score.hpp File Reference

```
#include <cstdint>
#include <unordered_map>
#include <vector>
#include <cl/fs/path.hpp>
#include "data_set_identifier.hpp"
```

Include dependency graph for distance_score.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cm

Functions

- std::uint64_t cm::distanceScore (const std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >> &segmentationPointsForConfig, const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t >> &manualSegmentationPoints)

7.52 confusion_matrix/include/fetch.hpp File Reference

```
#include <cl/exception.hpp>
```

Include dependency graph for fetch.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cm

Functions

- template<typename Map , typename Key >
  auto cm::fetch (const Map &map, const Key &key)

  Fetches a value from a map for a given key.

7.53 confusion_matrix/include/imu.hpp File Reference

```
#include <cstddef>
#include <array>
#include <iosfwd>
```

Include dependency graph for imu.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cm

Macros

- #define CM_IMU
- #define CM_IMU_X(enm) enm,
- #define CM_IMU_X(enm) +1
- #define CM_IMU_X(enm) ::cm::Imu::enm,

Enumerations

- enum cm::Imu { cm::Imu::CM_IMU_X, cm::Imu::CM_IMU }

  Scoped enum type for the IMUs.

Functions

- std::ostream & cm::operator<< (std::ostream &os, Imu imu)

  Prints `imu` *to* `os`.

Variables

- constexpr std::size_t cm::imuCount

  The amount of IMUs.
- constexpr std::array< Imu, imuCount > cm::imus

  An array of the IMU enumerators.

7.53.1 Macro Definition Documentation

7.53.1.1 CM_IMU

```
#define CM_IMU
```

Value:

```
CM_IMU_X(Accelerometer) \
CM_IMU_X(Gyroscope)
```

Definition at line 10 of file imu.hpp.

7.53.1.2 CM_IMU_X [1/3]

```
#define CM_IMU_X(
            enm ) enm,
```

Definition at line 18 of file imu.hpp.

7.53.1.3 CM_IMU_X [2/3]

```
#define CM_IMU_X(
            enm ) +1
```

Definition at line 18 of file imu.hpp.

7.53.1.4 CM_IMU_X [3/3]

```
#define CM_IMU_X(
            enm ) ::cm::Imu::enm,
```

Definition at line 18 of file imu.hpp.

7.54 confusion_matrix/include/interpolated_data_set_paths.hpp File Reference

```
#include <vector>
#include <cl/fs/path.hpp>
```

Include dependency graph for interpolated_data_set_paths.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cm

Functions

- std::vector< cl::fs::Path > cm::interpolatedDataSetPaths ()

  Returns the paths to the interpolated data sets.

7.55 confusion_matrix/include/manual_segmentation_point.hpp File Reference

```
#include <cstdint>
#include <iosfwd>
#include <unordered_map>
#include <vector>
#include <cl/exception.hpp>
#include "data_set_identifier.hpp"
```

Include dependency graph for manual_segmentation_point.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- class cm::ManualSegmentationPoint

  Type used to represent a manual segmentation point.

Namespaces

- cm

7.56 confusion_matrix/include/order_configurations_by_quality.hpp File Reference

```
#include <cstdint>
#include <iosfwd>
#include <unordered_map>
#include <vector>
#include <cl/fs/path.hpp>
#include "configuration.hpp"
#include "data_set_identifier.hpp"
```

Include dependency graph for order_configurations_by_quality.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- struct cm::ConfigWithDistanceScore

Namespaces

- cm

Functions

- bool cm::operator< (const ConfigWithDistanceScore &lhs, const ConfigWithDistanceScore &rhs) noexcept
- std::ostream & cm::operator<< (std::ostream &os, const ConfigWithDistanceScore &configWithDistScore)
- std::vector< ConfigWithDistanceScore > cm::orderConfigurationsByQuality (const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t >> &manualSegmentationPoints, const std::unordered_map< Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >>> &algorithmically DeterminedSegmentationPoints)

7.57 confusion_matrix/include/python_output.hpp File Reference

```
#include <cl/fs/path.hpp>
#include "configuration.hpp"
```

Include dependency graph for python_output.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cm

Functions

- std::string cm::pythonOutput (const cl::fs::Path &csvFilePath, const Configuration &segmentorConfiguration)

  Runs the Python segmentor on `path`.

7.58 confusion_matrix/include/segment.hpp File Reference

```
#include <cstdint>
#include <unordered_map>
#include <vector>
#include <cl/fs/path.hpp>
#include "configuration.hpp"
```

Include dependency graph for segment.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cm

Functions

- std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > cm::segment (const Configuration &segmentorConfiguration)

7.59 confusion_matrix/include/split_string.hpp File Reference

```
#include <string>
#include <vector>
#include <pl/string_view.hpp>
```

Include dependency graph for split_string.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cm

Functions

- std::vector< std::string > cm::splitString (std::string string, pl::string_view splitBy)

  Splits `string` *by* `splitBy`.

7.60 confusion_matrix/src/closest_one.cpp File Reference

```
#include <algorithm>
#include <tl/optional.hpp>
#include <cl/exception.hpp>
#include "closest_one.hpp"
```

```
#include "distance.hpp"
```

Include dependency graph for closest_one.cpp:

Namespaces

- cm

Functions

- std::uint64_t cm::closestOne (std::uint64_t algorithmicallyDeterminedSegmentationPoint, const std::vector< std::uint64_t > &manualSegmentationPoints)

  Finds the segmentation point in `manualSegmentationPoints` *that is the closest to* `algorithmically↩DeterminedSegmentationPoint`.

7.61 confusion_matrix/src/configuration.cpp File Reference

```
#include <algorithm>
#include <fstream>
#include <ostream>
#include <tuple>
#include <utility>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/begin_end_macro.hpp>
#include <cl/exception.hpp>
#include <cl/s2n.hpp>
#include "configuration.hpp"
#include "split_string.hpp"
```

Include dependency graph for configuration.cpp:

Namespaces

- cm

Macros

- #define CM_ENSURE_HAS_VALUE(dataMember)
- #define CM_ENSURE_CONTAINS(container, dataMember)

Functions

- bool cm::operator== (const Configuration &lhs, const Configuration &rhs) noexcept
- bool cm::operator< (const Configuration &lhs, const Configuration &rhs) noexcept
- std::ostream & cm::operator<< (std::ostream &os, const Configuration &config)

7.61.1 Macro Definition Documentation

7.61.1.1 CM_ENSURE_CONTAINS

```
#define CM_ENSURE_CONTAINS(
            container,
            dataMember )
```

Value:

```
  PL_BEGIN_MACRO                                                          \
  if (!contains(container, dataMember)) {                                 \
    CL_THROW_FMT(                                                         \
      "\"{}\" is not a valid option for \"{}\"", *dataMember, #dataMember); \
  }                                                                       \
  PL_END_MACRO
```

7.61.1.2 CM_ENSURE_HAS_VALUE

```
#define CM_ENSURE_HAS_VALUE(
            dataMember )
```

Value:

```
  PL_BEGIN_MACRO                                    \
  if (!dataMember.has_value()) {                    \
    CL_THROW_FMT("\"{}\" was nullopt!", #dataMember); \
  }                                                 \
  PL_END_MACRO
```

7.62 confusion_matrix/src/confusion_matrix.cpp File Reference

```
#include "confusion_matrix.hpp"
```

Include dependency graph for confusion_matrix.cpp:

Namespaces

- cm

7.63 confusion_matrix/src/confusion_matrix_best_configs.cpp File Reference

```
#include <algorithm>
#include <atomic>
#include <ostream>
#include <thread>
#include <type_traits>
#include <utility>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/meta/nested_types.hpp>
#include <pl/meta/remove_cvref.hpp>
#include <pl/numeric.hpp>
#include <pl/thd/thread_pool.hpp>
#include <cl/exception.hpp>
#include "confusion_matrix_best_configs.hpp"
#include "csv_file_info.hpp"
```

```
#include "fetch.hpp"
```

Include dependency graph for confusion_matrix_best_configs.cpp:

Namespaces

- cm

Functions

- std::ostream & cm::operator<< (std::ostream &os, const ConfigWithTotalConfusionMatrix &obj)
- std::vector< ConfigWithTotalConfusionMatrix > cm::confusionMatrixBestConfigs (const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t >> &manualSegmentationPoints, const std::unordered_map< Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >>> &algorithmically DeterminedSegmentationPoints, const std::function< bool(const ConfigWithTotalConfusionMatrix &, const ConfigWithTotalConfusionMatrix &)> &sorter)

  Determines the 'best' configurations.

7.64 confusion_matrix/src/create_segmentation_results.cpp File Reference

```
#include <future>
#include <thread>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/unreachable.hpp>
#include <cl/exception.hpp>
#include "create_segmentation_results.hpp"
#include "segment.hpp"
```

Include dependency graph for create_segmentation_results.cpp:

Namespaces

- cm

Functions

- std::unordered_map< cm::Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > > cm::createSegmentationResults ()

  Invokes Python to generate the segmentation points algorithmically.

7.65 confusion_matrix/src/csv_file_info.cpp File Reference

```
#include <csv.hpp>
#include <cl/exception.hpp>
#include <cl/s2n.hpp>
#include "csv_file_info.hpp"
```

Include dependency graph for csv_file_info.cpp:

Namespaces

- cm

7.66 confusion_matrix/src/data_set_identifier.cpp File Reference

```
#include <ostream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/string_view.hpp>
#include <pl/unreachable.hpp>
#include <cl/exception.hpp>
#include "data_set_identifier.hpp"
```

Include dependency graph for data_set_identifier.cpp:

Namespaces

- cm

Macros

- #define CM_DATA_SET_IDENTIFIER_X(enm) case DataSetIdentifier::enm: return #enm; /∗ stringify ∗/
- #define DSI DataSetIdentifier

Functions

- std::ostream & cm::operator<< (std::ostream &os, DataSetIdentifier dsi)

  Prints a DataSetIdentifier to an ostream.
- DataSetIdentifier cm::toDataSetIdentifier (const cl::fs::Path &path)

  Converts a path to a CSV file to the corresponding DataSetIdentifier.

7.66.1 Macro Definition Documentation

7.66.1.1 CM_DATA_SET_IDENTIFIER_X

```
#define CM_DATA_SET_IDENTIFIER_X(
            enm ) case DataSetIdentifier::enm:  return #enm; /* stringify */
```

7.66.1.2 DSI

```
#define DSI DataSetIdentifier
```

7.67 confusion_matrix/src/distance.cpp File Reference

```
#include <algorithm>
#include "distance.hpp"
```

Include dependency graph for distance.cpp:

Namespaces

- cm

Functions

- std::uint64_t cm::distance (std::uint64_t a, std::uint64_t b)

  Calculates the distance between `a` *and* `b`*.*

7.68 confusion_matrix/src/distance_score.cpp File Reference

```
#include "distance_score.hpp"
#include "closest_one.hpp"
#include "distance.hpp"
#include "fetch.hpp"
```

Include dependency graph for distance_score.cpp:

Namespaces

- cm

Functions

- std::uint64_t cm::distanceScore (const std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >> &segmentationPointsForConfig, const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_↩ t >> &manualSegmentationPoints)

7.69 confusion_matrix/src/imu.cpp File Reference

```
#include <ostream>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/inline.hpp>
#include <pl/unreachable.hpp>
#include "imu.hpp"
```

Include dependency graph for imu.cpp:

Namespaces

- cm

Macros

- #define CM_IMU_X(enm) case Imu::enm: return os << toLower(#enm);

Functions

- std::ostream & cm::operator<< (std::ostream &os, Imu imu)

  Prints `imu` *to* `os`.

7.69.1 Macro Definition Documentation

7.69.1.1 CM_IMU_X

```
#define CM_IMU_X(
            enm ) case Imu::enm:  return os << toLower(#enm);
```

7.70 confusion_matrix/src/interpolated_data_set_paths.cpp File Reference

```
#include <iterator>
#include <string>
#include <pl/algo/erase.hpp>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/string_view.hpp>
#include <cl/exception.hpp>
#include <cl/fs/directory_listing.hpp>
#include "interpolated_data_set_paths.hpp"
```

Include dependency graph for interpolated_data_set_paths.cpp:

Namespaces

- cm

Functions

- std::vector< cl::fs::Path > cm::interpolatedDataSetPaths ()

  Returns the paths to the interpolated data sets.

7.71 confusion_matrix/src/manual_segmentation_point.cpp File Reference

```
#include <cassert>
#include <cmath>
#include <functional>
#include <ostream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <csv.hpp>
```

```
#include <pl/algo/erase.hpp>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/numeric.hpp>
#include <pl/strcontains.hpp>
#include <cl/fs/path.hpp>
#include <cl/s2n.hpp>
#include "fetch.hpp"
#include "manual_segmentation_point.hpp"
```

Include dependency graph for manual_segmentation_point.cpp:

Namespaces

- cm

Macros

- #define DSI DataSetIdentifier

Functions

- bool cm::operator== (const ManualSegmentationPoint &lhs, const ManualSegmentationPoint &rhs) noexcept
- bool cm::operator!= (const ManualSegmentationPoint &lhs, const ManualSegmentationPoint &rhs) noexcept
- std::ostream & cm::operator<< (std::ostream &os, const ManualSegmentationPoint &manual↩ SegmentationPoint)

7.71.1 Macro Definition Documentation

7.71.1.1 DSI

```
#define DSI DataSetIdentifier
```

7.72 confusion_matrix/src/order_configurations_by_quality.cpp File Reference

```
#include <ostream>
#include <utility>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/algo/ranged_algorithms.hpp>
#include "distance_score.hpp"
#include "order_configurations_by_quality.hpp"
```

Include dependency graph for order_configurations_by_quality.cpp:

Namespaces

- cm

Functions

- bool cm::operator< (const ConfigWithDistanceScore &lhs, const ConfigWithDistanceScore &rhs) noexcept
- std::ostream & cm::operator<< (std::ostream &os, const ConfigWithDistanceScore &configWithDistScore)
- std::vector< ConfigWithDistanceScore > cm::orderConfigurationsByQuality (const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t >> &manualSegmentationPoints, const std::unordered_map< Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >>> &algorithmically DeterminedSegmentationPoints)

7.73 confusion_matrix/src/python_output.cpp File Reference

```
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/os.hpp>
#include <pl/string_view.hpp>
#include <cl/process.hpp>
#include "python_output.hpp"
```

Include dependency graph for python_output.cpp:

Namespaces

- cm

Macros

- #define CM_SEGMENTOR "./preprocessed_segment.sh"

  Object like macro for the segmentor script.
- #define CM_DEV_NULL "/dev/null"

  Object like macro for /dev/null.

Functions

- std::string cm::pythonOutput (const cl::fs::Path &csvFilePath, const Configuration &segmentorConfiguration)

  Runs the Python segmentor on `path`.

7.73.1 Macro Definition Documentation

7.73.1.1 CM_DEV_NULL

```
#define CM_DEV_NULL "/dev/null"
```

Object like macro for /dev/null.

Definition at line 24 of file python_output.cpp.

7.73.1.2 CM_SEGMENTOR

```
#define CM_SEGMENTOR "./preprocessed_segment.sh"
```

Object like macro for the segmentor script.

Definition at line 23 of file python_output.cpp.

7.74 confusion_matrix/src/segment.cpp File Reference

```
#include <cl/exception.hpp>
#include <cl/s2n.hpp>
#include "interpolated_data_set_paths.hpp"
#include "python_output.hpp"
#include "segment.hpp"
#include "split_string.hpp"
```

Include dependency graph for segment.cpp:

Namespaces

- cm

Functions

- std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > cm::segment (const Configuration &segmentorConfiguration)

7.75 confusion_matrix/src/split_string.cpp File Reference

```
#include <cstring>
#include <mutex>
#include "split_string.hpp"
```

Include dependency graph for split_string.cpp:

Namespaces

- cm

Functions

- std::vector< std::string > cm::splitString (std::string string, pl::string_view splitBy)

  Splits `string` *by* `splitBy`.

7.76 confusion_matrix/test/data_set_identifier_test.cpp File Reference

```
#include <array>
#include "gtest/gtest.h"
#include <cl/fs/path.hpp>
#include "data_set_identifier.hpp"
```

Include dependency graph for data_set_identifier_test.cpp:

Macros

- #define DSI ::cm::DataSetIdentifier

Functions

- TEST (DataSetIdentifier, shouldConvertPaths)

7.76.1 Macro Definition Documentation

7.76.1.1 DSI

```
#define DSI ::cm::DataSetIdentifier
```

7.76.2 Function Documentation

7.76.2.1 TEST()

```
TEST (
            DataSetIdentifier ,
            shouldConvertPaths  )
```

Definition at line 9 of file data_set_identifier_test.cpp.

Here is the call graph for this function:

7.77 confusion_matrix/test/interpolated_data_set_paths_test.cpp File Reference

```
#include <string>
#include "gtest/gtest.h"
#include <pl/algo/ranged_algorithms.hpp>
#include "interpolated_data_set_paths.hpp"
```

Include dependency graph for interpolated_data_set_paths_test.cpp:

Functions

- TEST (interpolatedDataSetPaths, shouldFetchPaths)

7.77.1 Function Documentation

7.77.1.1 TEST()

```
TEST (
            interpolatedDataSetPaths ,
            shouldFetchPaths  )
```

Definition at line 9 of file interpolated_data_set_paths_test.cpp.

Here is the call graph for this function:

7.78 confusion_matrix/test/manual_segmentation_point_test.cpp File Reference

```
#include <cstddef>
#include <array>
#include <ostream>
#include <sstream>
#include <string>
#include "gtest/gtest.h"
#include "manual_segmentation_point.hpp"
```

Include dependency graph for manual_segmentation_point_test.cpp:

Macros

- #define DSI ::cm::DataSetIdentifier

Functions

- TEST (ManualSegmentationPoint, shouldConstruct)
- TEST (ManualSegmentationPoint, shouldThrowWhenConstructingWithInvalidMinute)
- TEST (ManualSegmentationPoint, shouldThrowWhenConstructingWithInvalidSecond)
- TEST (ManualSegmentationPoint, shouldThrowWhenConstructingWithInvalidFrame)
- TEST (ManualSegmentationPoint, shouldConvertToMilliseconds)
- TEST (ManualSegmentationPoint, shouldConvertHourToMilliseconds)
- TEST (ManualSegmentationPoint, shouldConvertMinuteToMilliseconds)
- TEST (ManualSegmentationPoint, shouldConvertSecondToMilliseconds)
- TEST (ManualSegmentationPoint, shouldConvertFramesToMilliseconds)
- TEST (ManualSegmentationPoint, shouldBeAbleToImportCsvFile)
- TEST (ManualSegmentationPoint, shouldPrint)

7.78.1 Macro Definition Documentation

7.78.1.1 DSI

```
#define DSI ::cm::DataSetIdentifier
```

7.78.2 Function Documentation

7.78.2.1 TEST() [1/11]

```
TEST (
            ManualSegmentationPoint ,
            shouldBeAbleToImportCsvFile  )
```

Definition at line 98 of file manual_segmentation_point_test.cpp.

Here is the call graph for this function:

7.78.2.2 TEST() [2/11]

```
TEST (
            ManualSegmentationPoint ,
            shouldConstruct  )
```

Definition at line 12 of file manual_segmentation_point_test.cpp.

7.78.2.3 TEST() [3/11]

```
TEST (
            ManualSegmentationPoint ,
            shouldConvertFramesToMilliseconds  )
```

Definition at line 82 of file manual_segmentation_point_test.cpp.

7.78.2.4 TEST() [4/11]

```
TEST (
            ManualSegmentationPoint ,
            shouldConvertHourToMilliseconds  )
```

Definition at line 64 of file manual_segmentation_point_test.cpp.

7.78.2.5 TEST() [5/11]

```
TEST (
            ManualSegmentationPoint ,
            shouldConvertMinuteToMilliseconds  )
```

Definition at line 70 of file manual_segmentation_point_test.cpp.

7.78.2.6 TEST() [6/11]

```
TEST (
            ManualSegmentationPoint ,
            shouldConvertSecondToMilliseconds  )
```

Definition at line 76 of file manual_segmentation_point_test.cpp.

7.78.2.7 TEST() [7/11]

```
TEST (
            ManualSegmentationPoint ,
            shouldConvertToMilliseconds  )
```

Definition at line 58 of file manual_segmentation_point_test.cpp.

7.78.2.8 TEST() [8/11]

```
TEST (
            ManualSegmentationPoint ,
            shouldPrint  )
```

Definition at line 370 of file manual_segmentation_point_test.cpp.

7.78.2.9 TEST() [9/11]

```
TEST (
            ManualSegmentationPoint ,
            shouldThrowWhenConstructingWithInvalidFrame  )
```

Definition at line 46 of file manual_segmentation_point_test.cpp.

7.78.2.10 TEST() [10/11]

```
TEST (
            ManualSegmentationPoint ,
            shouldThrowWhenConstructingWithInvalidMinute  )
```

Definition at line 22 of file manual_segmentation_point_test.cpp.

7.78.2.11 TEST() [11/11]

```
TEST (
            ManualSegmentationPoint ,
            shouldThrowWhenConstructingWithInvalidSecond  )
```

Definition at line 34 of file manual_segmentation_point_test.cpp.

7.79 confusion_matrix/test/segment_test.cpp File Reference

```
#include <exception>
#include <stdexcept>
#include "gtest/gtest.h"
#include <pl/os.hpp>
#include "segment.hpp"
```

Include dependency graph for segment_test.cpp:

Macros

- #define EXPECT_SEGMENTATION_POINTS(path, ...) EXPECT_EQ((std::vector<std::uint64_t>{__VA_↩ ARGS__}), fetch(path))

Functions

- TEST (segment, shouldGetExpectedSegmentationPointsFromPython)

7.79.1 Macro Definition Documentation

7.79.1.1 EXPECT_SEGMENTATION_POINTS

```
#define EXPECT_SEGMENTATION_POINTS(
            path,
            ...  ) EXPECT_EQ((std::vector<std::uint64_t>{__VA_ARGS__}), fetch(path))
```

7.79.2 Function Documentation

7.79.2.1 TEST()

```
TEST (
            segment ,
            shouldGetExpectedSegmentationPointsFromPython  )
```

Definition at line 11 of file segment_test.cpp.

Here is the call graph for this function:

7.80 confusion_matrix/test/split_string_test.cpp File Reference

```
#include "gtest/gtest.h"
#include "split_string.hpp"
```

Include dependency graph for split_string_test.cpp:

Functions

- TEST (splitString, shouldSplitString)

7.80.1 Function Documentation

7.80.1.1 TEST()

```
TEST (
            splitString ,
            shouldSplitString  )
```

Definition at line 5 of file split_string_test.cpp.

Here is the call graph for this function:

7.81 counting/include/above_threshold.hpp File Reference

```
#include <vector>
#include "cl/data_point.hpp"
#include "cl/data_set.hpp"
```

Include dependency graph for above_threshold.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- ctg

Functions

- std::vector< cl::DataPoint > ctg::aboveThreshold (const cl::DataSet &dataSet, long double accelerometer↩ Threshold, long double gyroscopeThreshold)

7.82 counting/include/average_comparison_value_calculator.hpp File Reference

```
#include "cl/channel.hpp"
#include "cl/data_set.hpp"
#include "cl/sensor.hpp"
```

Include dependency graph for average_comparison_value_calculator.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- ctg

Functions

- long double ctg::averageComparisonValueCalculator (cl::Sensor sensor, cl::Channel channel, const cl::DataSet &dataSet)

7.83 counting/include/half_maximum_comparison_value_calculator.hpp File Reference

```
#include "cl/channel.hpp"
#include "cl/data_set.hpp"
```

```
#include "cl/sensor.hpp"
```

Include dependency graph for half_maximum_comparison_value_calculator.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- ctg

Functions

- long double ctg::halfMaximumComparisonValueCalculator (cl::Sensor sensor, cl::Channel channel, const cl::DataSet &dataSet)

7.84 counting/include/is_relevant.hpp File Reference

```
#include <cmath>
#include "cl/channel.hpp"
#include "cl/data_set.hpp"
```

```
#include "cl/sensor.hpp"
```

Include dependency graph for is_relevant.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- ctg

Functions

- template<typename ComparisonValueCalculator >
  bool ctg::isRelevant (cl::Sensor sensor, cl::Channel channel, const cl::DataSet &dataSet, ComparisonValue↩ Calculator comparisonValueCalculator)

7.85 counting/include/percentage_of.hpp File Reference

```
#include <cstddef>
```

Include dependency graph for percentage_of.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- ctg

Functions

- constexpr long double ctg::percentageOf (std::size_t amount, std::size_t totalCount) noexcept

7.86 counting/include/run_above_threshold.hpp File Reference

```
#include <iosfwd>
#include "cl/data_set.hpp"
```

Include dependency graph for run_above_threshold.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- ctg

Functions

- void ctg::runAboveThreshold (std::ostream &aboveThresholdLogOutputFileStream, const cl::DataSet &dataSet)

7.87 counting/src/above_threshold.cpp File Reference

```
#include <cinttypes>
#include <array>
#include <utility>
#include <tl/casts.hpp>
#include <pl/numeric.hpp>
```

```
#include "above_threshold.hpp"
```

Include dependency graph for above_threshold.cpp:

Namespaces

- ctg

Macros

- #define CL_CHANNEL_X(enm, v, accessor) {accessor, cl::Channel::enm},

Functions

- std::vector< cl::DataPoint > ctg::aboveThreshold (const cl::DataSet &dataSet, long double accelerometer↩ Threshold, long double gyroscopeThreshold)

7.87.1 Macro Definition Documentation

7.87.1.1 CL_CHANNEL_X

```
#define CL_CHANNEL_X(
            enm,
            v,
            accessor ) {accessor, cl::Channel::enm},
```

7.87.2 Variable Documentation

7.87.2.1 channel

```
cl::Channel channel
```

Definition at line 18 of file above_threshold.cpp.

7.87.2.2 channelAccessor

```
cl::DataSet::ChannelAccessor channelAccessor
```

Definition at line 17 of file above_threshold.cpp.

7.88 counting/src/average_comparison_value_calculator.cpp File Reference

```
#include <cassert>
#include <cstdio>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include "average_comparison_value_calculator.hpp"
```

Include dependency graph for average_comparison_value_calculator.cpp:

Namespaces

- ctg

Functions

- long double ctg::averageComparisonValueCalculator (cl::Sensor sensor, cl::Channel channel, const cl::DataSet &dataSet)

7.89 counting/src/half_maximum_comparison_value_calculator.cpp File Reference

```
#include <cassert>
#include <cstdio>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include "half_maximum_comparison_value_calculator.hpp"
```

Include dependency graph for half_maximum_comparison_value_calculator.cpp:

Namespaces

- ctg

Functions

- long double ctg::halfMaximumComparisonValueCalculator (cl::Sensor sensor, cl::Channel channel, const cl::DataSet &dataSet)

7.90 counting/src/run_above_threshold.cpp File Reference

```
#include <ostream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include "cl/channel.hpp"
#include "cl/data_point.hpp"
#include "above_threshold.hpp"
#include "percentage_of.hpp"
#include "run_above_threshold.hpp"
```

Include dependency graph for run_above_threshold.cpp:

Namespaces

- ctg

Functions

- void ctg::runAboveThreshold (std::ostream &aboveThresholdLogOutputFileStream, const cl::DataSet &dataSet)

7.91 counting/test/above_threshold_test.cpp File Reference

```
#include "gtest/gtest.h"
#include "above_threshold.hpp"
```

Include dependency graph for above_threshold_test.cpp:

Macros

- #define EXPECT_LONG_DOUBLE_EQ(a, b) EXPECT_DOUBLE_EQ(static_cast<double>(a), static_cast<double>(b))

Functions

- TEST (aboveThreshold, shouldFindDataPointsIfThereAreAny)

7.91.1 Macro Definition Documentation

7.91.1.1 EXPECT_LONG_DOUBLE_EQ

```
#define EXPECT_LONG_DOUBLE_EQ(
            a,
            b ) EXPECT_DOUBLE_EQ(static_cast<double>(a), static_cast<double>(b))
```

Definition at line 6 of file above_threshold_test.cpp.

7.91.2 Function Documentation

7.91.2.1 TEST()

```
TEST (
            aboveThreshold ,
            shouldFindDataPointsIfThereAreAny  )
```

Definition at line 10 of file above_threshold_test.cpp.

Here is the call graph for this function:

7.92 counting/test/percentage_of_test.cpp File Reference

```
#include "gtest/gtest.h"
#include "percentage_of.hpp"
```

Include dependency graph for percentage_of_test.cpp:

Macros

- #define EXPECT_LONG_DOUBLE_EQ(a, b) EXPECT_DOUBLE_EQ(static_cast<double>(a), static_↩cast<double>(b))

Functions

- TEST (percentageOf, shouldWork)

7.92.1 Macro Definition Documentation

7.92.1.1 EXPECT_LONG_DOUBLE_EQ

```
#define EXPECT_LONG_DOUBLE_EQ(
            a,
            b ) EXPECT_DOUBLE_EQ(static_cast<double>(a), static_cast<double>(b))
```

Definition at line 6 of file percentage_of_test.cpp.

7.92.2 Function Documentation

7.92.2.1 TEST()

```
TEST (
            percentageOf ,
            shouldWork  )
```

Definition at line 10 of file percentage_of_test.cpp.

Here is the call graph for this function:

7.93 csv_lib/include/cl/channel.hpp File Reference

```
#include <cstddef>
#include <cstdint>
#include <array>
#include <iosfwd>
#include "cl/data_set.hpp"
```

Include dependency graph for channel.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- struct cl::data_set_accessor< Chan >

  Maps a `Channel` to its associated accessor member function pointer of the `DataSet` type.

Namespaces

- cl

Macros

- #define CL_CHANNEL
- #define CL_CHANNEL_X(enumerator, value, dataSetAccessor) enumerator = value,
- #define CL_CHANNEL_X(enumerator, value, dataSetAccessor) +1
- #define CL_CHANNEL_X(enm, v, a) ::cl::Channel::enm,
- #define CL_CHANNEL_X(enumerator, value, dataSetAccessor)

Enumerations

- enum cl::Channel : std::uint64_t { cl::Channel::CL_CHANNEL_X, cl::Channel::CL_CHANNEL }

  Scoped enum for the 6 sensor channels.

Functions

- DataSet::ChannelAccessor cl::dataSetAccessor (Channel channel)

  Returns the data set accessor for the `Channel` given.
- std::ostream & cl::operator<< (std::ostream &os, Channel channel)

  Prints a given `Channel` to `os`.
- bool cl::isAccelerometer (Channel channel)

  Checks whether a given `Channel` is an accelerometer channel.
- bool cl::isGyroscope (Channel channel)

  Checks whether a given `Channel` is a gyroscope channel.
- long double cl::threshold (Channel channel)

  Returns the threshold value for `channel`.

Variables

- constexpr std::size_t cl::channelCount

  The amount of sensor channels (6).
- constexpr std::array< Channel, channelCount > cl::channels

  An array of the sensor channel enumerators.
- template<Channel Chan>
  constexpr CL_CHANNEL DataSet::ChannelAccessor cl::data_set_accessor_v = data_set_accessor<Chan>↩
  ::f
- constexpr long double cl::accelerometerThreshold {1.99L}

  Constant for the maximum acceptable accelerometer value (in positive and negative direction).
- constexpr long double cl::gyroscopeThreshold {1999.99L}

  Constant for the maximum acceptable gyroscope value (in positive and negative direction).

7.93.1 Macro Definition Documentation

7.93.1.1 CL_CHANNEL

```
#define CL_CHANNEL
```

Value:

```
CL_CHANNEL_X(AccelerometerX, 1, &::cl::DataSet::accelerometerX) \
CL_CHANNEL_X(AccelerometerY, 2, &::cl::DataSet::accelerometerY) \
CL_CHANNEL_X(AccelerometerZ, 3, &::cl::DataSet::accelerometerZ) \
CL_CHANNEL_X(GyroscopeX, 4, &::cl::DataSet::gyroscopeX)         \
CL_CHANNEL_X(GyroscopeY, 5, &::cl::DataSet::gyroscopeY)         \
CL_CHANNEL_X(GyroscopeZ, 6, &::cl::DataSet::gyroscopeZ)
```

Definition at line 11 of file channel.hpp.

7.93.1.2 CL_CHANNEL_X [1/4]

```
#define CL_CHANNEL_X(
              enm,
              v,
              a ) ::cl::Channel::enm,
```

Definition at line 55 of file channel.hpp.

7.93.1.3 CL_CHANNEL_X [2/4]

```
#define CL_CHANNEL_X(
              enumerator,
              value,
              dataSetAccessor ) enumerator = value,
```

Definition at line 55 of file channel.hpp.

7.93.1.4 CL_CHANNEL_X [3/4]

```
#define CL_CHANNEL_X(
              enumerator,
              value,
              dataSetAccessor ) +1
```

Definition at line 55 of file channel.hpp.

7.93.1.5 CL_CHANNEL_X [4/4]

```
#define CL_CHANNEL_X(
            enumerator,
            value,
            dataSetAccessor )
```

Value:

```
template<>                                                          \
struct data_set_accessor<Channel::enumerator> {                     \
  static constexpr ::cl::DataSet::ChannelAccessor f = dataSetAccessor; \
};
```

Definition at line 55 of file channel.hpp.

7.94 csv_lib/include/cl/column.hpp File Reference

Contains definitions regarding the columns of the 'old' CSV files, i.e., the CSV files that have not been preprocessed using MATLAB.

```
#include <cstddef>
#include <cstdint>
#include "cl/sensor.hpp"
```

Include dependency graph for column.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- struct cl::col_traits< Col >

  Column traits for the columns of the old, non-preprocessed CSV files. Includes the index (0 based) of the column and the C++ data type to be used for the cell contents in that column.

Namespaces

- cl

Macros

- #define CL_SPECIALIZE_COL_TRAITS(column, columnType)

Typedefs

- template<Column Col>
  using cl::column_type = typename col_traits< Col >::type

  Meta function for the type to use for the given Column.

Enumerations

- enum cl::Column : std::size_t {
  cl::Column::Time, cl::Column::HardwareTimestamp, cl::Column::ExtractId, cl::Column::Trigger, cl::Column::AccelerometerX, cl::Column::AccelerometerY, cl::Column::AccelerometerZ, cl::Column::GyroscopeX, cl::Column::GyroscopeY, cl::Column::GyroscopeZ, cl::Column::SamplingRate }

  A scoped enum for the columns of the old, non-preprocessed CSV files.

Functions

- cl::CL_SPECIALIZE_COL_TRAITS (Column::Time, long double)
- cl::CL_SPECIALIZE_COL_TRAITS (Column::HardwareTimestamp, std::uint64_t)
- cl::CL_SPECIALIZE_COL_TRAITS (Column::ExtractId, Sensor)
- cl::CL_SPECIALIZE_COL_TRAITS (Column::Trigger, std::uint64_t)
- cl::CL_SPECIALIZE_COL_TRAITS (Column::AccelerometerX, long double)
- cl::CL_SPECIALIZE_COL_TRAITS (Column::AccelerometerY, long double)
- cl::CL_SPECIALIZE_COL_TRAITS (Column::AccelerometerZ, long double)
- cl::CL_SPECIALIZE_COL_TRAITS (Column::GyroscopeX, long double)
- cl::CL_SPECIALIZE_COL_TRAITS (Column::GyroscopeY, long double)
- cl::CL_SPECIALIZE_COL_TRAITS (Column::GyroscopeZ, long double)
- cl::CL_SPECIALIZE_COL_TRAITS (Column::SamplingRate, std::uint64_t)

Variables

- template<Column Col>
  constexpr std::size_t cl::column_index = col_traits<Col>::index

  Meta function for the 0 based index of the given Column.

7.94.1 Detailed Description

Contains definitions regarding the columns of the 'old' CSV files, i.e., the CSV files that have not been preprocessed using MATLAB.

7.94.2 Macro Definition Documentation

7.94.2.1 CL_SPECIALIZE_COL_TRAITS

```
#define CL_SPECIALIZE_COL_TRAITS(
            column,
            columnType )
```

Value:

```
template<>                                                          \
struct col_traits<column> {                                         \
  static constexpr std::size_t index = static_cast<std::size_t>(column); \
  using type                         = columnType;                \
}
```

Definition at line 40 of file column.hpp.

7.95 csv_lib/include/cl/data_point.hpp File Reference

```
#include <iosfwd>
#include <string>
#include "cl/channel.hpp"
#include "cl/sensor.hpp"
```

Include dependency graph for data_point.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- class cl::DataPoint

  Type to represent a data point in a data set.

Namespaces

- cl

7.96 csv_lib/include/cl/data_set.hpp File Reference

```
#include <cstddef>
#include <array>
#include <string>
#include <vector>
#include "cl/column.hpp"
#include "cl/error.hpp"
#include "cl/sensor.hpp"
```

Include dependency graph for data_set.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- class cl::DataSet

Namespaces

- cl

7.97 csv_lib/include/cl/dos2unix.hpp File Reference

```
#include <cstddef>
#include <vector>
#include <pl/byte.hpp>
```

Include dependency graph for dos2unix.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cl

Functions

- std::vector< pl::byte > cl::dos2unix (const void *p, std::size_t size)

  Converts DOS / Microsoft Windows line endings to UNIX line endings.

7.98 csv_lib/include/cl/error.hpp File Reference

```
#include <cstddef>
#include <iosfwd>
#include <string>
#include <tl/expected.hpp>
#include <pl/current_function.hpp>
#include "cl/to_string.hpp"
```

Include dependency graph for error.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- class cl::Error

Namespaces

- cl

Macros

- #define CL_ERROR_KIND
- #define CL_ERROR_KIND_X(kind) kind,
- #define CL_UNEXPECTED(kind, message)

Typedefs

- template<typename Ty >
  using cl::Expected = tl::expected< Ty, Error >

7.98.1 Macro Definition Documentation

7.98.1.1 CL_ERROR_KIND

```
#define CL_ERROR_KIND
```

Value:

```
CL_ERROR_KIND_X(Filesystem)       \
CL_ERROR_KIND_X(InvalidArgument) \
CL_ERROR_KIND_X(OutOfRange)       \
CL_ERROR_KIND_X(Parsing)          \
CL_ERROR_KIND_X(Logic)            \
CL_ERROR_KIND_X(OperatingSystem)
```

Definition at line 14 of file error.hpp.

7.98.1.2 CL_ERROR_KIND_X

```
#define CL_ERROR_KIND_X(
              kind ) kind,
```

Definition at line 27 of file error.hpp.

7.98.1.3 CL_UNEXPECTED

```
#define CL_UNEXPECTED(
              kind,
              message )
```

Value:

```
::tl::make_unexpected(            \
  ::cl::Error{kind, __FILE__, PL_CURRENT_FUNCTION, __LINE__, message})
```

Definition at line 67 of file error.hpp.

7.99 csv_lib/include/cl/exception.hpp File Reference

```
#include <cstddef>
#include <stdexcept>
#include <string>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/current_function.hpp>
```

Include dependency graph for exception.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- class cl::Exception

Namespaces

- cl

Macros

- #define CL_THROW(what_arg) throw ::cl::Exception { __FILE__, PL_CURRENT_FUNCTION, __LINE__↩ , what_arg }
- #define CL_THROW_FMT(fmt_str, ...) CL_THROW(::fmt::format(fmt_str, __VA_ARGS__))

7.99.1 Macro Definition Documentation

7.99.1.1 CL_THROW

```
#define CL_THROW(
            what_arg ) throw ::cl::Exception { __FILE__, PL_CURRENT_FUNCTION, __LINE__↩
, what_arg }
```

Definition at line 42 of file exception.hpp.

7.99.1.2 CL_THROW_FMT

```
#define CL_THROW_FMT(
            fmt_str,
            ...  ) CL_THROW(::fmt::format(fmt_str, __VA_ARGS__))
```

Definition at line 45 of file exception.hpp.

7.100 csv_lib/include/cl/fs/directory_listing.hpp File Reference

```
#include <vector>
#include <cl/error.hpp>
#include <cl/fs/path.hpp>
```

Include dependency graph for directory_listing.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cl
- cl::fs

Enumerations

- enum cl::fs::DirectoryListingOption { cl::fs::DirectoryListingOption::None, cl::fs::DirectoryListingOption::ExcludeDotAndDotDot }

  Options for directoryListing.

Functions

- Expected< std::vector< Path > > cl::fs::directoryListing (const Path &directoryPath, DirectoryListingOption directoryListingOption=DirectoryListingOption::ExcludeDotAndDotDot)

  Creates a listing of the contents of a directory.

7.101 csv_lib/include/cl/fs/file.hpp File Reference

```
#include <cstdint>
#include "cl/fs/path.hpp"
```

Include dependency graph for file.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- class cl::fs::File

  Represents a file.

Namespaces

- cl
- cl::fs

7.102 csv_lib/include/cl/fs/file_stream.hpp File Reference

```
#include <cstddef>
#include <cstdint>
#include <cstdio>
#include <vector>
```

```
#include <pl/byte.hpp>
#include <pl/noncopyable.hpp>
#include "cl/error.hpp"
#include "cl/fs/file.hpp"
```

Include dependency graph for file_stream.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- class cl::fs::FileStream

  A binary file stream.

Namespaces

- cl
- cl::fs

7.103 csv_lib/include/cl/fs/path.hpp File Reference

```
#include <functional>
#include <iosfwd>
#include <string>
#include <pl/annotations.hpp>
```

`#include <pl/total_order.hpp>`
Include dependency graph for path.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- class cl::fs::Path

  A filesystem path.
- struct std::hash<::cl::fs::Path >

Namespaces

- cl
- cl::fs

7.104 csv_lib/include/cl/fs/separator.hpp File Reference

`#include <pl/os.hpp>`
Include dependency graph for separator.hpp:

This graph shows which files directly or indirectly include this file:

Macros

- #define CL_FS_SEPARATOR "\\"

  The filesystem separator of the operating system.

7.104.1 Macro Definition Documentation

7.104.1.1 CL_FS_SEPARATOR

```
#define CL_FS_SEPARATOR "\\"
```

The filesystem separator of the operating system.

Definition at line 11 of file separator.hpp.

7.105 csv_lib/include/cl/fs/windows.hpp File Reference

Contains Microsoft Windows specific functions.

```
#include <string>
#include <pl/os.hpp>
#include <pl/string_view.hpp>
```

```
#include <Windows.h>
```

Include dependency graph for windows.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cl
- cl::fs

Functions

- std::wstring cl::fs::utf8ToUtf16 (pl::string_view utf8)

  Converts a UTF-8 encoded string to a UTF-16 encoded wstring.
- std::string cl::fs::utf16ToUtf8 (pl::wstring_view utf16)

  Converts a UTF-16 encoded wide character string to UTF-8 string.
- std::wstring cl::fs::formatError (DWORD errorCode)

  Formats a WINAPI error code to a UTF-16 encoded wide character string.

7.105.1 Detailed Description

Contains Microsoft Windows specific functions.

7.106 csv_lib/include/cl/process.hpp File Reference

```
#include <cstdio>
#include <pl/noncopyable.hpp>
#include <pl/string_view.hpp>
#include "error.hpp"
```

Include dependency graph for process.hpp:

This graph shows which files directly or indirectly include this file:

Classes

- class cl::Process

Namespaces

- cl

7.107 csv_lib/include/cl/read_csv_file.hpp File Reference

```
#include <string>
#include <vector>
#include <pl/string_view.hpp>
```

```
#include "cl/error.hpp"
```

Include dependency graph for read_csv_file.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cl

Enumerations

- enum cl::CsvFileKind { cl::CsvFileKind::Raw, cl::CsvFileKind::Fixed }

Functions

- Expected< std::vector< std::vector< std::string > > > cl::readCsvFile (pl::string_view csvFilePath, std↩ ::vector< std::string > *columnNames=nullptr, CsvFileKind csvFileKind=CsvFileKind::Fixed) noexcept

7.108 csv_lib/include/cl/s2n.hpp File Reference

```
#include <cassert>
#include <cstddef>
#include <string>
#include <type_traits>
#include <fmt/format.h>
#include <fmt/ostream.h>
```

```
#include "cl/error.hpp"
```

Include dependency graph for s2n.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cl

Functions

- template<typename Integer >
  Expected< Integer > cl::s2n (const std::string &str, std::size_t *pos=nullptr, [[maybe_unused]] int base=10)

7.109 csv_lib/include/cl/sensor.hpp File Reference

```
#include <cstdint>
#include <array>
#include <iosfwd>
```

Include dependency graph for sensor.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cl

Macros

- #define CL_SENSOR
- #define CL_SENSOR_X(enumerator, value) enumerator = value,
- #define CL_SENSOR_X(enm, v) ::cl::Sensor::enm,

Enumerations

- enum cl::Sensor : std::uint64_t { cl::Sensor::CL_SENSOR_X, cl::Sensor::CL_SENSOR }

Functions

- std::ostream & cl::operator<< (std::ostream &os, Sensor sensor)

Variables

- constexpr std::array< Sensor, 4 > cl::sensors

7.109.1 Macro Definition Documentation

7.109.1.1 CL_SENSOR

```
#define CL_SENSOR
```

Value:

```
CL_SENSOR_X(LeftArm, 769)   \
CL_SENSOR_X(Belly, 770)     \
CL_SENSOR_X(RightArm, 771)  \
CL_SENSOR_X(Chest, 772)
```

Definition at line 9 of file sensor.hpp.

7.109.1.2 CL_SENSOR_X [1/2]

```
#define CL_SENSOR_X(
            enm,
            v ) ::cl::Sensor::enm,
```

Definition at line 16 of file sensor.hpp.

7.109.1.3 CL_SENSOR_X [2/2]

```
#define CL_SENSOR_X(
            enumerator,
            value ) enumerator = value,
```

Definition at line 16 of file sensor.hpp.

7.110 csv_lib/include/cl/to_string.hpp File Reference

```
#include <ios>
#include <locale>
#include <sstream>
#include <string>
#include <type_traits>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/meta/remove_cvref.hpp>
```

Include dependency graph for to_string.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- cl

Functions

- template<typename Ty >
  std::string cl::to_string (const Ty &ty)

7.111 csv_lib/include/cl/use_unbuffered_io.hpp File Reference

This graph shows which files directly or indirectly include this file:

Namespaces

- cl

Functions

- void cl::useUnbufferedIo ()

7.112 csv_lib/src/cl/channel.cpp File Reference

```
#include <cassert>
#include <cinttypes>
#include <cstring>
#include <ostream>
#include <tl/casts.hpp>
#include <pl/numeric.hpp>
#include <pl/unreachable.hpp>
#include "cl/channel.hpp"
```

Include dependency graph for channel.cpp:

Namespaces

- cl

Macros

- #define CL_CHANNEL_X(enm, v, acc) case Channel::enm: return data_set_accessor_v<Channel::enm>;
- #define CL_CHANNEL_X(enumerator, value, dataSetAccessor) case Channel::enumerator: return os << #enumerator;

Functions

- DataSet::ChannelAccessor cl::dataSetAccessor (Channel channel)

  Returns the data set accessor for the `Channel` given.
- std::ostream & cl::operator<< (std::ostream &os, Channel channel)

  Prints a given `Channel` to `os`.
- bool cl::isAccelerometer (Channel channel)

  Checks whether a given `Channel` is an accelerometer channel.
- bool cl::isGyroscope (Channel channel)

  Checks whether a given `Channel` is a gyroscope channel.
- long double cl::threshold (Channel channel)

  Returns the threshold value for `channel`.

7.112.1 Macro Definition Documentation

7.112.1.1 CL_CHANNEL_X [1/2]

```
#define CL_CHANNEL_X(
            enm,
            v,
            acc ) case Channel::enm:  return data_set_accessor_v<Channel::enm>;
```

7.112.1.2 CL_CHANNEL_X [2/2]

```
#define CL_CHANNEL_X(
            enumerator,
            value,
            dataSetAccessor ) case Channel::enumerator:  return os << #enumerator;
```

7.113 csv_lib/src/cl/data_point.cpp File Reference

```
#include <ostream>
#include <utility>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include "cl/data_point.hpp"
```

Include dependency graph for data_point.cpp:

Namespaces

- cl

Functions

- std::ostream & cl::operator<< (std::ostream &os, const DataPoint &dataPoint)
- dataPoint fileName ()
- dataPoint dataPoint time ()
- dataPoint dataPoint dataPoint sensor ()
- dataPoint dataPoint dataPoint dataPoint channel ()
- dataPoint dataPoint dataPoint dataPoint dataPoint value ())

7.113.1 Function Documentation

7.113.1.1 channel()

```
dataPoint dataPoint dataPoint dataPoint channel ( )
```

7.113.1.2 fileName()

```
dataPoint fileName ( )
```

Here is the caller graph for this function:

7.113.1.3 sensor()

```
dataPoint dataPoint dataPoint sensor ( )
```

Here is the caller graph for this function:

7.113.1.4 time()

```
dataPoint dataPoint time ( )
```

7.113.1.5 value()

```
dataPoint dataPoint dataPoint dataPoint dataPoint value ( )
```

Here is the caller graph for this function:

7.114 csv_lib/src/cl/data_set.cpp File Reference

```
#include <cassert>
#include <cmath>
#include <cstdint>
#include <stdexcept>
#include <string>
#include <type_traits>
#include <utility>
#include <fmt/format.h>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/assert.hpp>
#include "cl/data_set.hpp"
#include "cl/sensor.hpp"
```

Include dependency graph for data_set.cpp:

Namespaces

- cl

7.115 csv_lib/src/cl/dos2unix.cpp File Reference

```
#include "cl/dos2unix.hpp"
```

Include dependency graph for dos2unix.cpp:

Namespaces

- cl

Functions

- std::vector< pl::byte > cl::dos2unix (const void *p, std::size_t size)

  Converts DOS / Microsoft Windows line endings to UNIX line endings.

7.116 csv_lib/src/cl/error.cpp File Reference

```
#include <cassert>
#include <ostream>
#include <utility>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/string_view.hpp>
#include "cl/error.hpp"
#include "cl/exception.hpp"
```

Include dependency graph for error.cpp:

Namespaces

- cl

Macros

- #define CL_ERROR_KIND_X(kind) case Error::kind: return #kind;

Functions

- std::ostream & cl::operator<< (std::ostream &os, const Error &error)

7.116.1 Macro Definition Documentation

7.116.1.1 CL_ERROR_KIND_X

```
#define CL_ERROR_KIND_X(
              kind ) case Error::kind:  return #kind;
```

7.117 csv_lib/src/cl/exception.cpp File Reference

```
#include <utility>
#include "cl/exception.hpp"
```

Include dependency graph for exception.cpp:

Namespaces

- cl

7.118 csv_lib/src/cl/fs/directory_listing.cpp File Reference

```
#include <cassert>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/algo/erase.hpp>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/os.hpp>
#include <cl/fs/windows.hpp>
#include <dirent.h>
#include <sys/types.h>
#include <cl/fs/directory_listing.hpp>
```

Include dependency graph for directory_listing.cpp:

Namespaces

- cl
- cl::fs

Functions

- Expected< std::vector< Path > > cl::fs::directoryListing (const Path &directoryPath, DirectoryListingOption directoryListingOption=DirectoryListingOption::ExcludeDotAndDotDot)

  Creates a listing of the contents of a directory.

7.119 csv_lib/src/cl/fs/file.cpp File Reference

```
#include <cassert>
#include <cstdio>
#include <string>
#include <utility>
#include <pl/os.hpp>
#include "cl/fs/windows.hpp"
#include <Windows.h>
#include "cl/fs/file.hpp"
```

Include dependency graph for file.cpp:

Namespaces

- cl
- cl::fs

7.120 csv_lib/src/cl/fs/file_stream.cpp File Reference

```
#include <cassert>
#include <pl/os.hpp>
#include <pl/unreachable.hpp>
#include "cl/fs/windows.hpp"
#include <wchar.h>
#include <fmt/format.h>
#include <fmt/ostream.h>
```

```
#include "cl/fs/file_stream.hpp"
```

Include dependency graph for file_stream.cpp:

Namespaces

- cl
- cl::fs

7.121 csv_lib/src/cl/fs/path.cpp File Reference

```
#include <cassert>
#include <cstring>
#include <ostream>
#include <utility>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/os.hpp>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include "cl/fs/path.hpp"
```

Include dependency graph for path.cpp:

Namespaces

- cl
- cl::fs

Functions

- std::ostream & cl::fs::operator<< (std::ostream &os, const Path &path)
- bool cl::fs::operator< (const Path &lhs, const Path &rhs) noexcept
- bool cl::fs::operator== (const Path &lhs, const Path &rhs) noexcept

7.122 csv_lib/src/cl/fs/windows.cpp File Reference

```
#include <cassert>
#include <cstddef>
#include "cl/fs/windows.hpp"
```

Include dependency graph for windows.cpp:

Namespaces

- cl
- cl::fs

Functions

- std::wstring cl::fs::utf8ToUtf16 (pl::string_view utf8)

  Converts a UTF-8 encoded string to a UTF-16 encoded wstring.
- std::string cl::fs::utf16ToUtf8 (pl::wstring_view utf16)

  Converts a UTF-16 encoded wide character string to UTF-8 string.
- std::wstring cl::fs::formatError (DWORD errorCode)

  Formats a WINAPI error code to a UTF-16 encoded wide character string.

7.123 csv_lib/src/cl/process.cpp File Reference

```
#include <stdio.h>
#include <cassert>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/os.hpp>
```

```
#include "cl/process.hpp"
```

Include dependency graph for process.cpp:

Namespaces

- cl

7.124 csv_lib/src/cl/read_csv_file.cpp File Reference

```
#include <algorithm>
#include <string>
#include <vector>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <csv.hpp>
#include <pl/unreachable.hpp>
#include "cl/read_csv_file.hpp"
```

Include dependency graph for read_csv_file.cpp:

Namespaces

- cl

Functions

- Expected< std::vector< std::vector< std::string > > > cl::readCsvFile (pl::string_view csvFilePath, std↩ ::vector< std::string > *columnNames=nullptr, CsvFileKind csvFileKind=CsvFileKind::Fixed) noexcept

7.125 csv_lib/src/cl/sensor.cpp File Reference

```
#include <cassert>
#include <ostream>
#include "cl/sensor.hpp"
```

Include dependency graph for sensor.cpp:

Namespaces

- cl

Macros

- #define CL_SENSOR_X(enumerator, value) case Sensor::enumerator: return os << #enumerator;

Functions

- std::ostream & cl::operator<< (std::ostream &os, Sensor sensor)

7.125.1 Macro Definition Documentation

7.125.1.1 CL_SENSOR_X

```
#define CL_SENSOR_X(
            enumerator,
            value ) case Sensor::enumerator:  return os << #enumerator;
```

7.126 csv_lib/src/cl/use_unbuffered_io.cpp File Reference

```
#include <cstdio>
#include <ios>
#include <iostream>
#include "cl/use_unbuffered_io.hpp"
```

Include dependency graph for use_unbuffered_io.cpp:

Namespaces

- cl

Functions

- void cl::useUnbufferedIo ()

7.127 csv_lib/test/channel_test.cpp File Reference

```
#include <sstream>
#include "gtest/gtest.h"
#include "channel.hpp"
```

Include dependency graph for channel_test.cpp:

Functions

- TEST (channel, shouldHaveCorrectCount)
- TEST (channel, shouldHaveCorrectValues)
- TEST (channel, shouldPrintCorrectly)
- TEST (channel, shouldMapToCorrectDataSetAccessors)

7.127.1 Function Documentation

7.127.1.1 TEST() [1/4]

```
TEST (
            channel ,
            shouldHaveCorrectCount  )
```

Definition at line 7 of file channel_test.cpp.

7.127.1.2 TEST() [2/4]

```
TEST (
            channel ,
            shouldHaveCorrectValues  )
```

Definition at line 9 of file channel_test.cpp.

7.127.1.3 TEST() [3/4]

```
TEST (
            channel ,
            shouldMapToCorrectDataSetAccessors  )
```

Definition at line 35 of file channel_test.cpp.

Here is the call graph for this function:

7.127.1.4 TEST() [4/4]

```
TEST (
            channel ,
            shouldPrintCorrectly  )
```

Definition at line 19 of file channel_test.cpp.

7.128 csv_lib/test/column_test.cpp File Reference

```
#include <type_traits>
#include "gtest/gtest.h"
```

```
#include "column.hpp"
```

Include dependency graph for column_test.cpp:

Functions

- TEST (column, shouldHaveCorrectIndex)
- TEST (column, shouldHaveCorrectColumnType)

7.128.1 Function Documentation

7.128.1.1 TEST() [1/2]

```
TEST (
            column ,
            shouldHaveCorrectColumnType  )
```

Definition at line 22 of file column_test.cpp.

7.128.1.2 TEST() [2/2]

```
TEST (
            column ,
            shouldHaveCorrectIndex  )
```

Definition at line 7 of file column_test.cpp.

7.129 csv_lib/test/data_point_test.cpp File Reference

```
#include <sstream>
#include "gtest/gtest.h"
#include "data_point.hpp"
```

Include dependency graph for data_point_test.cpp:

Functions

- TEST (DataPoint, shouldPrintCorrectly)
- TEST (DataPoint, shouldGetValuesCorrectly)

Variables

- const cl::DataPoint dp

7.129.1 Function Documentation

7.129.1.1 TEST() [1/2]

```
TEST (
            DataPoint ,
            shouldGetValuesCorrectly  )
```

Definition at line 23 of file data_point_test.cpp.

Here is the call graph for this function:

7.129.1.2 TEST() [2/2]

```
TEST (
            DataPoint ,
            shouldPrintCorrectly  )
```

Definition at line 14 of file data_point_test.cpp.

7.129.2 Variable Documentation

7.129.2.1 dp

```
const cl::DataPoint dp
```

Initial value:

```
{
   "file.csv",
   0.01,
   cl::Sensor::Chest,
   cl::Channel::AccelerometerX,
   50.01}
```

Definition at line 7 of file data_point_test.cpp.

7.130 csv_lib/test/data_set_test.cpp File Reference

```
#include "gtest/gtest.h"
#include <pl/string_view.hpp>
#include "data_set.hpp"
#include "read_csv_file.hpp"
```

Include dependency graph for data_set_test.cpp:

Macros

- #define EXPECT_LONG_DOUBLE_EQ(a, b) EXPECT_DOUBLE_EQ(static_cast<double>(a), static_↩cast<double>(b))

Functions

- TEST (DataSet, shouldBeAbleToCreateFromValidData)
- TEST (DataSet, shouldNotBeAbleToCreateFromEmtyMatrix)
- TEST (DataSet, shouldNotBeAbleToCreateFromJaggedMatrix)
- TEST (DataSet, shouldNotBeAbleToCreateFromInvalidData)

7.130.1 Macro Definition Documentation

7.130.1.1 EXPECT_LONG_DOUBLE_EQ

```
#define EXPECT_LONG_DOUBLE_EQ(
            a,
            b ) EXPECT_DOUBLE_EQ(static_cast<double>(a), static_cast<double>(b))
```

7.130.2 Function Documentation

7.130.2.1 TEST() [1/4]

```
TEST (
            DataSet ,
            shouldBeAbleToCreateFromValidData  )
```

Definition at line 17 of file data_set_test.cpp.

Here is the call graph for this function:

7.130.2.2 TEST() [2/4]

```
TEST (
            DataSet ,
            shouldNotBeAbleToCreateFromEmtyMatrix  )
```

Definition at line 68 of file data_set_test.cpp.

Here is the call graph for this function:

7.130.2.3 TEST() [3/4]

```
TEST (
            DataSet ,
            shouldNotBeAbleToCreateFromInvalidData  )
```

Definition at line 108 of file data_set_test.cpp.

Here is the call graph for this function:

7.130.2.4 TEST() [4/4]

```
TEST (
            DataSet ,
            shouldNotBeAbleToCreateFromJaggedMatrix  )
```

Definition at line 80 of file data_set_test.cpp.

Here is the call graph for this function:

7.131 csv_lib/test/directory_listing_test.cpp File Reference

```
#include "gtest/gtest.h"
#include <pl/string_view.hpp>
#include <cl/fs/directory_listing.hpp>
#include <iostream>
```

Include dependency graph for directory_listing_test.cpp:

Functions

- TEST (directoryListing, shouldFindFiles)
- TEST (directoryListing, shouldFindFilesWithDotAndDotDot)
- TEST (directoryListing, shouldReturnErrorWhenPathDoesNotExist)

7.131.1 Function Documentation

7.131.1.1 TEST() [1/3]

```
TEST (
            directoryListing ,
            shouldFindFiles  )
```

Definition at line 13 of file directory_listing_test.cpp.

Here is the call graph for this function:

7.131.1.2 TEST() [2/3]

```
TEST (
            directoryListing ,
            shouldFindFilesWithDotAndDotDot  )
```

Definition at line 28 of file directory_listing_test.cpp.

Here is the call graph for this function:

7.131.1.3 TEST() [3/3]

```
TEST (
            directoryListing ,
            shouldReturnErrorWhenPathDoesNotExist  )
```

Definition at line 46 of file directory_listing_test.cpp.

Here is the call graph for this function:

7.132 csv_lib/test/error_test.cpp File Reference

```
#include <cstring>
#include <sstream>
#include "gtest/gtest.h"
#include <pl/string_view.hpp>
#include "error.hpp"
#include "exception.hpp"
```

Include dependency graph for error_test.cpp:

Functions

- TEST (error, shouldPrint)
- TEST (error, shouldReturnValues)
- TEST (error, shouldThrowExceptionWhenRaiseIsCalled)
- TEST (error, shouldCreateExpectedWithUnexpected)

Variables

- const cl::Error error

7.132.1 Function Documentation

7.132.1.1 TEST() [1/4]

```
TEST (
            error ,
            shouldCreateExpectedWithUnexpected  )
```

Definition at line 59 of file error_test.cpp.

7.132.1.2 TEST() [2/4]

```
TEST (
            error ,
            shouldPrint  )
```

Definition at line 19 of file error_test.cpp.

7.132.1.3 TEST() [3/4]

```
TEST (
            error ,
            shouldReturnValues  )
```

Definition at line 29 of file error_test.cpp.

7.132.1.4 TEST() [4/4]

```
TEST (
            error ,
            shouldThrowExceptionWhenRaiseIsCalled  )
```

Definition at line 37 of file error_test.cpp.

7.132.2 Variable Documentation

7.132.2.1 error

```
const cl::Error error
```

Initial value:

```
{
   cl::Error::Filesystem,
   "test_file.cpp",
   "bad_function",
   48,
   "Couldn't initialize the flux capacitor."}
```

Definition at line 12 of file error_test.cpp.

7.133 csv_lib/test/exception_test.cpp File Reference

```
#include "gtest/gtest.h"
#include "exception.hpp"
```

Include dependency graph for exception_test.cpp:

Functions

- TEST (exception, shouldWork)

7.133.1 Function Documentation

7.133.1.1 TEST()

```
TEST (
            exception ,
            shouldWork  )
```

Definition at line 5 of file exception_test.cpp.

7.134 csv_lib/test/read_csv_file_test.cpp File Reference

```
#include "gtest/gtest.h"
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/string_view.hpp>
#include "read_csv_file.hpp"
```

Include dependency graph for read_csv_file_test.cpp:

Functions

- TEST (readCsvFile, shouldReadCsvFile)
- TEST (readCsvFile, shouldNotReadNonexistantCsvFile)

7.134.1 Function Documentation

7.134.1.1 TEST() [1/2]

```
TEST (
            readCsvFile ,
            shouldNotReadNonexistantCsvFile  )
```

Definition at line 30 of file read_csv_file_test.cpp.

Here is the call graph for this function:

7.134.1.2 TEST() [2/2]

```
TEST (
            readCsvFile ,
            shouldReadCsvFile  )
```

Definition at line 8 of file read_csv_file_test.cpp.

Here is the call graph for this function:

7.135 csv_lib/test/s2n_test.cpp File Reference

```
#include "gtest/gtest.h"
#include <pl/string_view.hpp>
```

```
#include "cl/s2n.hpp"
```

Include dependency graph for s2n_test.cpp:

Functions

- TEST (s2n, shouldWork)
- TEST (s2n, shouldReturnInvalidArgumentErrorIfInputIsInvalid)
- TEST (s2n, shouldReturnOutOfRangeErrorIfInputIsOutOfRange)

7.135.1 Function Documentation

7.135.1.1 TEST() [1/3]

```
TEST (
            s2n ,
            shouldReturnInvalidArgumentErrorIfInputIsInvalid  )
```

Definition at line 21 of file s2n_test.cpp.

7.135.1.2 TEST() [2/3]

```
TEST (
            s2n ,
            shouldReturnOutOfRangeErrorIfInputIsOutOfRange  )
```

Definition at line 29 of file s2n_test.cpp.

7.135.1.3 TEST() [3/3]

```
TEST (
            s2n ,
            shouldWork  )
```

Definition at line 7 of file s2n_test.cpp.

Here is the call graph for this function:

7.136 csv_lib/test/sensor_test.cpp File Reference

```
#include <sstream>
#include "gtest/gtest.h"
#include "sensor.hpp"
```

Include dependency graph for sensor_test.cpp:

Functions

- TEST (sensor, shouldHaveCorrectValues)
- TEST (sensor, shouldPrintCorretly)

7.136.1 Function Documentation

7.136.1.1 TEST() [1/2]

```
TEST (
            sensor ,
            shouldHaveCorrectValues  )
```

Definition at line 7 of file sensor_test.cpp.

7.136.1.2 TEST() [2/2]

```
TEST (
            sensor ,
            shouldPrintCorretly  )
```

Definition at line 15 of file sensor_test.cpp.

Here is the call graph for this function:

7.137 csv_lib/test/to_string_test.cpp File Reference

```
#include "gtest/gtest.h"
#include "channel.hpp"
#include "data_point.hpp"
#include "error.hpp"
#include "sensor.hpp"
#include "to_string.hpp"
```

Include dependency graph for to_string_test.cpp:

Functions

- TEST (to_string, test)

7.137.1 Function Documentation

7.137.1.1 TEST()

```
TEST (
            to_string ,
            test  )
```

Definition at line 9 of file to_string_test.cpp.

Here is the call graph for this function:

7.138 fix_csv/include/adjust_hardware_timestamp.hpp File Reference

```
#include <cstdint>
#include <string>
```

Include dependency graph for adjust_hardware_timestamp.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- fmc

Functions

- void fmc::adjustHardwareTimestamp (std::string *cellContent, const std::string &nextRowHardware↩ Timestamp, std::uint64_t *overflowCount)

7.139 fix_csv/include/convert_to_unix_line_endings.hpp File Reference

```
#include <string>
```

Include dependency graph for convert_to_unix_line_endings.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- fmc

Functions

- bool fmc::convertToUnixLineEndings (const std::string &csvPath)

7.140 fix_csv/include/create_backup_file.hpp File Reference

```
#include <string>
```

Include dependency graph for create_backup_file.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- fmc

Functions

- bool fmc::createBackupFile (const std::string &csvFilePath, const std::string &backupFilePath)

7.141 fix_csv/include/delete_non_bosch_sensors.hpp File Reference

```
#include <string>
#include <vector>
```

Include dependency graph for delete_non_bosch_sensors.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- fmc

Functions

- void fmc::deleteNonBoschSensors (std::vector< std::vector< std::string >> *data)

7.142 fix_csv/include/delete_out_of_bounds_values.hpp File Reference

```
#include <string>
#include <vector>
#include "cl/error.hpp"
```

Include dependency graph for delete_out_of_bounds_values.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- fmc

Functions

- cl::Expected< void > fmc::deleteOutOfBoundsValues (std::vector< std::vector< std::string >> *data)

7.143 fix_csv/include/remove_zeros_from_field.hpp File Reference

`#include <string>`
Include dependency graph for remove_zeros_from_field.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- fmc

Functions

- void fmc::removeZerosFromField (std::string *field)

7.144 fix_csv/include/restore_from_backup.hpp File Reference

```
#include <string>
```

Include dependency graph for restore_from_backup.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- fmc

Functions

- bool fmc::restoreFromBackup (const std::string &csvFilePath, const std::string &backupFilePath)

7.145 fix_csv/include/write_file.hpp File Reference

```
#include <string>
#include <vector>
#include <pl/string_view.hpp>
```

Include dependency graph for write_file.hpp:

This graph shows which files directly or indirectly include this file:

Namespaces

- fmc

Functions

- bool fmc::writeFile (pl::string_view csvPath, pl::string_view csvFileExtension, const std::vector< std::string > &columnNames, const std::vector< std::vector< std::string >> &data)

7.146 fix_csv/src/adjust_hardware_timestamp.cpp File Reference

```
#include <climits>
#include <cstdint>
#include <cstdio>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/current_function.hpp>
#include "cl/s2n.hpp"
#include "cl/to_string.hpp"
#include "adjust_hardware_timestamp.hpp"
```

Include dependency graph for adjust_hardware_timestamp.cpp:

Namespaces

- fmc

Functions

- void fmc::adjustHardwareTimestamp (std::string *cellContent, const std::string &nextRowHardware↩ Timestamp, std::uint64_t *overflowCount)

7.147 fix_csv/src/convert_to_unix_line_endings.cpp File Reference

```
#include <cstdio>
#include <vector>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/byte.hpp>
#include "cl/dos2unix.hpp"
#include "cl/error.hpp"
#include "cl/fs/file.hpp"
#include "cl/fs/file_stream.hpp"
#include "convert_to_unix_line_endings.hpp"
```

Include dependency graph for convert_to_unix_line_endings.cpp:

Namespaces

- fmc

Functions

- bool fmc::convertToUnixLineEndings (const std::string &csvPath)

7.148 fix_csv/src/create_backup_file.cpp File Reference

```
#include "cl/fs/file.hpp"
#include "create_backup_file.hpp"
```

Include dependency graph for create_backup_file.cpp:

Namespaces

- fmc

Functions

- bool fmc::createBackupFile (const std::string &csvFilePath, const std::string &backupFilePath)

7.149 fix_csv/src/delete_non_bosch_sensors.cpp File Reference

```
#include <tl/casts.hpp>
#include <pl/algo/erase.hpp>
#include <pl/string_view.hpp>
#include <pl/stringify.hpp>
#include <pl/unreachable.hpp>
#include "cl/column.hpp"
#include "cl/sensor.hpp"
#include "delete_non_bosch_sensors.hpp"
```

Include dependency graph for delete_non_bosch_sensors.cpp:

Namespaces

- fmc

Macros

- #define CL_SENSOR_X(enm, value) case cl::Sensor::enm: return PL_STRINGIFY(value);

Functions

- void fmc::deleteNonBoschSensors (std::vector< std::vector< std::string >> *data)

7.149.1 Macro Definition Documentation

7.149.1.1 CL_SENSOR_X

```
#define CL_SENSOR_X(
            enm,
            value ) case cl::Sensor::enm:  return PL_STRINGIFY(value);
```

7.150 fix_csv/src/delete_out_of_bounds_values.cpp File Reference

```
#include <cmath>
#include <array>
#include <stdexcept>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/algo/erase.hpp>
#include <pl/algo/ranged_algorithms.hpp>
#include "cl/channel.hpp"
#include "cl/column.hpp"
#include "cl/s2n.hpp"
#include "delete_out_of_bounds_values.hpp"
```

Include dependency graph for delete_out_of_bounds_values.cpp:

Namespaces

- fmc

Functions

- cl::Expected< void > fmc::deleteOutOfBoundsValues (std::vector< std::vector< std::string >> *data)

7.151 fix_csv/src/remove_zeros_from_field.cpp File Reference

```
#include <algorithm>
#include "remove_zeros_from_field.hpp"
```

Include dependency graph for remove_zeros_from_field.cpp:

Namespaces

- fmc

Functions

- void fmc::removeZerosFromField (std::string *field)

7.152 fix_csv/src/restore_from_backup.cpp File Reference

```
#include <cstdio>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include "cl/fs/file.hpp"
```

```
#include "restore_from_backup.hpp"
```

Include dependency graph for restore_from_backup.cpp:

Namespaces

- fmc

Functions

- bool fmc::restoreFromBackup (const std::string &csvFilePath, const std::string &backupFilePath)

7.153 fix_csv/src/write_file.cpp File Reference

```
#include <fstream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <csv.hpp>
#include "convert_to_unix_line_endings.hpp"
#include "write_file.hpp"
```

Include dependency graph for write_file.cpp:

Namespaces

- fmc

Functions

- bool fmc::writeFile (pl::string_view csvPath, pl::string_view csvFileExtension, const std::vector< std::string > &columnNames, const std::vector< std::vector< std::string >> &data)

7.154 fix_csv/test/adjust_hardware_timestamp_test.cpp File Reference

```
#include <cstdint>
#include <cstring>
#include <string>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include "gtest/gtest.h"
#include "cl/to_string.hpp"
#include "adjust_hardware_timestamp.hpp"
```

Include dependency graph for adjust_hardware_timestamp_test.cpp:

Functions

- TEST (adjustHardwareTimestamp, shouldDoNothingForNonOverflowedValue)
- TEST (adjustHardwareTimestamp, shouldIncrementOverflowCount)
- TEST (adjustHardwareTimestamp, shouldWorkForOneRoundOfOverflow)
- TEST (adjustHardwareTimestamp, shouldWorkForTwoRoundsOfOverflow)
- TEST (adjustHardwareTimestamp, shouldWork)

7.154.1 Function Documentation

7.154.1.1 TEST() [1/5]

```
TEST (
            adjustHardwareTimestamp ,
            shouldDoNothingForNonOverflowedValue  )
```

Definition at line 15 of file adjust_hardware_timestamp_test.cpp.

Here is the call graph for this function:

7.154.1.2 TEST() [2/5]

```
TEST (
            adjustHardwareTimestamp ,
            shouldIncrementOverflowCount  )
```

Definition at line 26 of file adjust_hardware_timestamp_test.cpp.

Here is the call graph for this function:

7.154.1.3 TEST() [3/5]

```
TEST (
            adjustHardwareTimestamp ,
            shouldWork  )
```

Definition at line 132 of file adjust_hardware_timestamp_test.cpp.

Here is the call graph for this function:

7.154.1.4 TEST() [4/5]

```
TEST (
            adjustHardwareTimestamp ,
            shouldWorkForOneRoundOfOverflow  )
```

Definition at line 48 of file adjust_hardware_timestamp_test.cpp.

Here is the call graph for this function:

7.154.1.5 TEST() [5/5]

```
TEST (
            adjustHardwareTimestamp ,
            shouldWorkForTwoRoundsOfOverflow  )
```

Definition at line 96 of file adjust_hardware_timestamp_test.cpp.

Here is the call graph for this function:

7.155 fix_csv/test/remove_zeros_from_field_test.cpp File Reference

```
#include "gtest/gtest.h"
#include "remove_zeros_from_field.hpp"
```

Include dependency graph for remove_zeros_from_field_test.cpp:

Functions

- TEST (removeZerosFromField, shouldRemoveDotAndZeros)
- TEST (removeZerosFromField, shouldNotRemoveIfNonZerosFollow)
- TEST (removeZerosFromField, shouldNotRemoveIfNoDot)
- TEST (removeZerosFromField, shouldDoNothingIfStringIsEmpty)
- TEST (removeZerosFromField, shouldDeleteStringIfStringIsSingleDot)
- TEST (removeZerosFromField, shouldDeleteStringIfStringIsDotAndZero)

7.155.1 Function Documentation

7.155.1.1 TEST() [1/6]

```
TEST (
            removeZerosFromField ,
            shouldDeleteStringIfStringIsDotAndZero  )
```

Definition at line 53 of file remove_zeros_from_field_test.cpp.

Here is the call graph for this function:

7.155.1.2 TEST() [2/6]

```
TEST (
            removeZerosFromField ,
            shouldDeleteStringIfStringIsSingleDot  )
```

Definition at line 44 of file remove_zeros_from_field_test.cpp.

Here is the call graph for this function:

7.155.1.3 TEST() [3/6]

```
TEST (
            removeZerosFromField ,
            shouldDoNothingIfStringIsEmpty  )
```

Definition at line 35 of file remove_zeros_from_field_test.cpp.

Here is the call graph for this function:

7.155.1.4 TEST() [4/6]

```
TEST (
            removeZerosFromField ,
            shouldNotRemoveIfNoDot  )
```

Definition at line 25 of file remove_zeros_from_field_test.cpp.

Here is the call graph for this function:

7.155.1.5 TEST() [5/6]

```
TEST (
            removeZerosFromField ,
            shouldNotRemoveIfNonZerosFollow  )
```

Definition at line 15 of file remove_zeros_from_field_test.cpp.

Here is the call graph for this function:

7.155.1.6 TEST() [6/6]

```
TEST (
            removeZerosFromField ,
            shouldRemoveDotAndZeros  )
```

Definition at line 5 of file remove_zeros_from_field_test.cpp.

Here is the call graph for this function:

Index

~FileStream
  cl::fs::FileStream, 153
~Process
  cl::Process, 181

above_threshold.cpp
  channel, 284
  channelAccessor, 285
  CL_CHANNEL_X, 284
above_threshold_test.cpp
  EXPECT_LONG_DOUBLE_EQ, 287
  TEST, 288
aboveThreshold
  ctg, 64
accelerometerAverage
  cl::DataSet, 128
accelerometerMaximum
  cl::DataSet, 129
accelerometerThreshold
  cl, 24
AccelerometerX
  cl, 14
accelerometerX
  cl::DataSet, 130
AccelerometerY
  cl, 14
accelerometerY
  cl::DataSet, 130
AccelerometerZ
  cl, 14
accelerometerZ
  cl::DataSet, 130
addTrueSubtractFalseSorter
  cm, 51
adjust_hardware_timestamp_test.cpp
  TEST, 359–361
adjustHardwareTimestamp
  fmc, 69
asMilliseconds
  cm::ManualSegmentationPoint, 169
averageComparisonValueCalculator
  ctg, 64

base_type
  cl::Exception, 143
Both
  cs, 54
build
  cm::Configuration::Builder, 76
  cs::CsvLineBuilder, 110
Builder
  cm::Configuration, 96
  cm::Configuration::Builder, 76
Butterworth
  cs, 54

Channel
  cl, 13
channel
  above_threshold.cpp, 284
  cl::DataPoint, 124
  data_point.cpp, 314
channel.cpp
  CL_CHANNEL_X, 313
channel.hpp
  CL_CHANNEL, 292
  CL_CHANNEL_X, 292
channel_test.cpp
  TEST, 326, 327
ChannelAccessor
  cl::DataSet, 128
channelAccessor
  above_threshold.cpp, 285
channelCount
  cl, 24
channels
  cl, 24
cl, 11
  accelerometerThreshold, 24
  AccelerometerX, 14
  AccelerometerY, 14
  AccelerometerZ, 14
  Channel, 13
  channelCount, 24
  channels, 24
  CL_CHANNEL, 13
  CL_CHANNEL_X, 13
  CL_SENSOR, 14
  CL_SENSOR_X, 14
  CL_SPECIALIZE_COL_TRAITS, 15, 16
  Column, 13
  column_index, 25
  column_type, 13
  CsvFileKind, 14
  data_set_accessor_v, 25
  dataSetAccessor, 16
  dos2unix, 17
  Expected, 13
  ExtractId, 14
  Fixed, 14

gyroscopeThreshold, 25
GyroscopeX, 14
GyroscopeY, 14
GyroscopeZ, 14
HardwareTimestamp, 14
isAccelerometer, 18
isGyroscope, 18
operator<<, 19–21
Raw, 14
readCsvFile, 21
s2n, 22
SamplingRate, 14
Sensor, 14
sensors, 25
threshold, 22
Time, 14
to_string, 23
Trigger, 14
useUnbufferedIo, 24
cl::col_traits< Col >, 84
cl::data_set_accessor< Chan >, 121
cl::DataPoint, 122
channel, 124
DataPoint, 123
fileName, 124
operator<<, 127
sensor, 125
time, 125
value, 126
cl::DataSet, 127
accelerometerAverage, 128
accelerometerMaximum, 129
accelerometerX, 130
accelerometerY, 130
accelerometerZ, 130
ChannelAccessor, 128
create, 131
extractId, 132
fileName, 133
gyroscopeAverage, 133
gyroscopeMaximum, 134
gyroscopeX, 135
gyroscopeY, 135
gyroscopeZ, 135
hardwareTimestamp, 136
rowCount, 136
size_type, 128
time, 137
trigger, 137
cl::Error, 138
CL_ERROR_KIND, 139
Error, 139
file, 139
function, 139
Kind, 138
kind, 140
line, 140
message, 140
operator<<, 141
raise, 141
to_string, 141
cl::Exception, 142
base_type, 143
Exception, 143
file, 143
function, 144
line, 144
cl::fs, 26
directoryListing, 27
DirectoryListingOption, 26
ExcludeDotAndDotDot, 27
formatError, 28
None, 27
operator<, 28
operator<<, 29
operator==, 29
utf16ToUtf8, 29
utf8ToUtf16, 30
cl::fs::File, 145
copyTo, 146
create, 147
exists, 147
File, 146
moveTo, 148
path, 149
remove, 149
size, 150
cl::fs::FileStream, 151
~FileStream, 153
create, 153
FileStream, 153
OpenMode, 152
operator=, 154
PL_NONCOPYABLE, 155
Read, 153
readAll, 155
ReadWrite, 153
this_type, 152
Write, 153
write, 155
cl::fs::Path, 175
exists, 176
isDirectory, 177
isFile, 178
operator<, 179
operator<<, 180
operator==, 180
Path, 176
str, 178
cl::Process, 181
~Process, 181
create, 182
file, 182
operator=, 182
PL_NONCOPYABLE, 182
Process, 181

this_type, 181
CL_CHANNEL
channel.hpp, 292
cl, 13
CL_CHANNEL_X
above_threshold.cpp, 284
channel.cpp, 313
channel.hpp, 292
cl, 13
CL_ERROR_KIND
cl::Error, 139
error.hpp, 298
CL_ERROR_KIND_X
error.cpp, 318
error.hpp, 299
CL_FS_SEPARATOR
separator.hpp, 305
CL_SENSOR
cl, 14
sensor.hpp, 310
CL_SENSOR_X
cl, 14
delete_non_bosch_sensors.cpp, 356
sensor.cpp, 324
sensor.hpp, 310, 311
CL_SPECIALIZE_COL_TRAITS
cl, 15, 16
column.hpp, 295
CL_THROW
exception.hpp, 300
CL_THROW_FMT
exception.hpp, 300
CL_UNEXPECTED
error.hpp, 299
closestOne
cm, 34
cm, 31
addTrueSubtractFalseSorter, 51
closestOne, 34
CM_DATA_SET_IDENTIFIER, 33
CM_DATA_SET_IDENTIFIER_X, 33
CM_IMU, 34
CM_IMU_X, 34
CM_SORTER, 35
confusionMatrixBestConfigs, 35
createSegmentationResults, 37
DataSetIdentifier, 33
disregardTrueNegativesSorter, 51
distance, 38
distanceScore, 39
fetch, 40
Imu, 33
imuCount, 51
imus, 52
interpolatedDataSetPaths, 41
operator!=, 42
operator<, 42
operator<<, 43–45
operator==, 46
orderConfigurationsByQuality, 46
pythonOutput, 47
segment, 48
splitString, 50
toDataSetIdentifier, 50
cm::Configuration, 85
Builder, 96
Configuration, 86
createFilePath, 87
deleteTooClose, 87
deleteTooCloseOptions, 87
deleteTooLowVariance, 88
deleteTooLowVarianceOptions, 88
filterKind, 89
filterKindOptions, 89
importSegmentationPoints, 90
imu, 91
imuOptions, 91
isInitialized, 92
operator<, 96
operator<<, 97
operator==, 97
segmentationKind, 92
segmentationKindOptions, 93
serializeSegmentationPoints, 93
skipWindow, 94
skipWindowOptions, 94
std::hash< Configuration >, 97
windowSize, 95
windowSizeOptions, 95
cm::Configuration::Builder, 75
build, 76
Builder, 76
deleteTooClose, 77
deleteTooLowVariance, 78
filterKind, 79
imu, 80
segmentationKind, 81
skipWindow, 82
windowSize, 83
cm::ConfigWithDistanceScore, 98
config, 99
ConfigWithDistanceScore, 98
distScore, 99
cm::ConfigWithTotalConfusionMatrix, 99
config, 101
ConfigWithTotalConfusionMatrix, 100
matrix, 101
operator<<, 101
cm::ConfusionMatrix, 102
ConfusionMatrix, 103
falseNegatives, 103
falsePositives, 103
incrementFalseNegatives, 103
incrementFalsePositives, 104
incrementTrueNegatives, 104
incrementTruePositives, 105

operator+=, 105
this_type, 102
totalCount, 106
trueNegatives, 106
truePositives, 106
cm::CsvFileInfo, 107
CsvFileInfo, 107
hardwareTimestamps, 108
cm::ManualSegmentationPoint, 167
asMilliseconds, 169
convertToHardwareTimestamps, 169
frame, 170
hour, 171
ManualSegmentationPoint, 168
minute, 171
operator!=, 173
operator<<, 174
operator==, 174
readCsvFile, 172
second, 173
CM_DATA_SET_IDENTIFIER
cm, 33
data_set_identifier.hpp, 244
CM_DATA_SET_IDENTIFIER_X
cm, 33
data_set_identifier.cpp, 261
data_set_identifier.hpp, 244
CM_DEV_NULL
python_output.cpp, 267
CM_ENSURE_CONTAINS
configuration.cpp, 257
CM_ENSURE_HAS_VALUE
configuration.cpp, 257
CM_IMU
cm, 34
imu.hpp, 248
CM_IMU_X
cm, 34
imu.cpp, 263
imu.hpp, 249
CM_SEGMENTOR
python_output.cpp, 267
CM_SORTER
cm, 35
confusion_matrix_best_configs.hpp, 240
CMakeLists.txt
include, 185–189
set, 185–189
Column
cl, 13
column.hpp
CL_SPECIALIZE_COL_TRAITS, 295
column_index
cl, 25
column_test.cpp
TEST, 328
column_type
cl, 13
compare_segmentation/CMakeLists.txt, 185
compare_segmentation/include/csv_line.hpp, 190
compare_segmentation/include/data_set_info.hpp, 191
compare_segmentation/include/filter_kind.hpp, 193
compare_segmentation/include/log_files.hpp, 194
compare_segmentation/include/log_info.hpp, 194
compare_segmentation/include/log_line.hpp, 195
compare_segmentation/include/mode.hpp, 196
compare_segmentation/include/paths.hpp, 198
compare_segmentation/include/segmentation_kind.hpp, 199
compare_segmentation/src/csv_line.cpp, 200
compare_segmentation/src/data_set_info.cpp, 200
compare_segmentation/src/filter_kind.cpp, 201
compare_segmentation/src/log_files.cpp, 201
compare_segmentation/src/log_info.cpp, 202
compare_segmentation/src/log_line.cpp, 203
compare_segmentation/src/main.cpp, 203
compare_segmentation/src/mode.cpp, 215
compare_segmentation/src/segmentation_kind.cpp, 217
compare_segmentation/test/CMakeLists.txt, 185
compare_segmentation/test/csv_line_test.cpp, 217
compare_segmentation/test/data_set_info_test.cpp, 218
compare_segmentation/test/log_files_test.cpp, 219
compare_segmentation/test/log_info_test.cpp, 221
compare_segmentation/test/log_line_test.cpp, 231
compare_segmentation/test/main.cpp, 205
compare_segmentation/test/mode_test.cpp, 233
config
cm::ConfigWithDistanceScore, 99
cm::ConfigWithTotalConfusionMatrix, 101
Configuration
cm::Configuration, 86
configuration.cpp
CM_ENSURE_CONTAINS, 257
CM_ENSURE_HAS_VALUE, 257
ConfigWithDistanceScore
cm::ConfigWithDistanceScore, 98
ConfigWithTotalConfusionMatrix
cm::ConfigWithTotalConfusionMatrix, 100
confusion_matrix/CMakeLists.txt, 189
confusion_matrix/include/closest_one.hpp, 235
confusion_matrix/include/configuration.hpp, 236
confusion_matrix/include/confusion_matrix.hpp, 237
confusion_matrix/include/confusion_matrix_best_configs.hpp, 238
confusion_matrix/include/create_segmentation_results.hpp, 240
confusion_matrix/include/csv_file_info.hpp, 241
confusion_matrix/include/data_set_identifier.hpp, 242
confusion_matrix/include/distance.hpp, 244
confusion_matrix/include/distance_score.hpp, 245
confusion_matrix/include/fetch.hpp, 246
confusion_matrix/include/imu.hpp, 247
confusion_matrix/include/interpolated_data_set_paths.hpp, 250

confusion_matrix/include/manual_segmentation_point.hpp, 251
confusion_matrix/include/order_configurations_by_quality.hpp, 252
confusion_matrix/include/python_output.hpp, 253
confusion_matrix/include/segment.hpp, 254
confusion_matrix/include/split_string.hpp, 255
confusion_matrix/src/closest_one.cpp, 255
confusion_matrix/src/configuration.cpp, 256
confusion_matrix/src/confusion_matrix.cpp, 258
confusion_matrix/src/confusion_matrix_best_configs.cpp, 258
confusion_matrix/src/create_segmentation_results.cpp, 259
confusion_matrix/src/csv_file_info.cpp, 260
confusion_matrix/src/data_set_identifier.cpp, 260
confusion_matrix/src/distance.cpp, 262
confusion_matrix/src/distance_score.cpp, 262
confusion_matrix/src/imu.cpp, 263
confusion_matrix/src/interpolated_data_set_paths.cpp, 264
confusion_matrix/src/main.cpp, 212
confusion_matrix/src/manual_segmentation_point.cpp, 264
confusion_matrix/src/order_configurations_by_quality.cpp, 266
confusion_matrix/src/python_output.cpp, 266
confusion_matrix/src/segment.cpp, 267
confusion_matrix/src/split_string.cpp, 268
confusion_matrix/test/CMakeLists.txt, 189
confusion_matrix/test/data_set_identifier_test.cpp, 269
confusion_matrix/test/interpolated_data_set_paths_test.cpp, 270
confusion_matrix/test/main.cpp, 214
confusion_matrix/test/manual_segmentation_point_test.cpp, 271
confusion_matrix/test/segment_test.cpp, 275
confusion_matrix/test/split_string_test.cpp, 276
confusion_matrix_best_configs.hpp
    CM_SORTER, 240
ConfusionMatrix
    cm::ConfusionMatrix, 103
confusionMatrixBestConfigs
    cm, 35
convertToHardwareTimestamps
    cm::ManualSegmentationPoint, 169
convertToUnixLineEndings
    fmc, 69
copyTo
    cl::fs::File, 146
counting/CMakeLists.txt, 186
counting/include/above_threshold.hpp, 278
counting/include/average_comparison_value_calculator.hpp, 279
counting/include/half_maximum_comparison_value_calculator.hpp, 279
counting/include/is_relevant.hpp, 280
counting/include/percentage_of.hpp, 281
counting/include/run_above_threshold.hpp, 282
counting/src/above_threshold.cpp, 283
counting/src/average_comparison_value_calculator.cpp, 285
counting/src/half_maximum_comparison_value_calculator.cpp, 286
counting/src/main.cpp, 206
counting/src/run_above_threshold.cpp, 286
counting/test/above_threshold_test.cpp, 287
counting/test/CMakeLists.txt, 186
counting/test/main.cpp, 208
counting/test/percentage_of_test.cpp, 289
create
    cl::DataSet, 131
    cl::fs::File, 147
    cl::fs::FileStream, 153
    cl::Process, 182
    cs::LogInfo, 158
createBackupFile
    fmc, 70
createFilePath
    cm::Configuration, 87
createSegmentationResults
    cm, 37
cs, 52
    Both, 54
    Butterworth, 54
    CS_MODE, 54
    CS_MODE_X, 54
    CS_SPECIALIZE_DATA_SET_INFO, 54–58
    FilterKind, 53
    logFiles, 58
    logPath, 63
    Maxima, 54
    Minima, 54
    Mode, 54
    MovingAverage, 54
    oldLogPath, 63
    operator!=, 59
    operator<<, 59, 60
    operator==, 61
    parseMode, 61
    PL_DEFINE_EXCEPTION_TYPE, 62
    repetitionCount, 62
    SegmentationKind, 54
cs::CsvLineBuilder, 108
    build, 110
    CsvLineBuilder, 110
    dataSet, 110
    deleteLowVariance, 111
    deleteTooClose, 112
    filter, 113
    isOld, 114
    kind, 115
    repetitions, 116
    segmentationPoints, 117
    sensor, 118
    skipWindow, 119

this_type, 109
windowSize, 120
cs::data_set_info< Tag >, 122
cs::LogInfo, 157
create, 158
deleteLowVariance, 159
deleteTooClose, 159
filterKind, 159
invalidSensor, 163
isInitialized, 160
logFilePath, 160
LogInfo, 158
operator!=, 162
operator<<, 162
operator==, 163
segmentationKind, 160
sensor, 161
skipWindow, 161
windowSize, 161
cs::LogLine, 163
fileName, 164
filePath, 165
invalidSensor, 167
parse, 165
segmentationPointCount, 166
sensor, 166
CS_MODE
cs, 54
mode.hpp, 197
CS_MODE_X
cs, 54
mode.cpp, 216
mode.hpp, 197
CS_SPECIALIZE_DATA_SET_INFO
cs, 54–58
data_set_info.hpp, 192
csv_lib/CMakeLists.txt, 187
csv_lib/include/cl/channel.hpp, 290
csv_lib/include/cl/column.hpp, 293
csv_lib/include/cl/data_point.hpp, 295
csv_lib/include/cl/data_set.hpp, 296
csv_lib/include/cl/dos2unix.hpp, 297
csv_lib/include/cl/error.hpp, 298
csv_lib/include/cl/exception.hpp, 299
csv_lib/include/cl/fs/directory_listing.hpp, 301
csv_lib/include/cl/fs/file.hpp, 302
csv_lib/include/cl/fs/file_stream.hpp, 302
csv_lib/include/cl/fs/path.hpp, 303
csv_lib/include/cl/fs/separator.hpp, 304
csv_lib/include/cl/fs/windows.hpp, 305
csv_lib/include/cl/process.hpp, 307
csv_lib/include/cl/read_csv_file.hpp, 307
csv_lib/include/cl/s2n.hpp, 308
csv_lib/include/cl/sensor.hpp, 309
csv_lib/include/cl/to_string.hpp, 311
csv_lib/include/cl/use_unbuffered_io.hpp, 312
csv_lib/src/cl/channel.cpp, 312
csv_lib/src/cl/data_point.cpp, 314
csv_lib/src/cl/data_set.cpp, 317
csv_lib/src/cl/dos2unix.cpp, 317
csv_lib/src/cl/error.cpp, 318
csv_lib/src/cl/exception.cpp, 319
csv_lib/src/cl/fs/directory_listing.cpp, 319
csv_lib/src/cl/fs/file.cpp, 320
csv_lib/src/cl/fs/file_stream.cpp, 320
csv_lib/src/cl/fs/path.cpp, 321
csv_lib/src/cl/fs/windows.cpp, 322
csv_lib/src/cl/process.cpp, 322
csv_lib/src/cl/read_csv_file.cpp, 323
csv_lib/src/cl/sensor.cpp, 324
csv_lib/src/cl/use_unbuffered_io.cpp, 325
csv_lib/test/channel_test.cpp, 325
csv_lib/test/CMakeLists.txt, 187
csv_lib/test/column_test.cpp, 327
csv_lib/test/data_point_test.cpp, 329
csv_lib/test/data_set_test.cpp, 331
csv_lib/test/directory_listing_test.cpp, 335
csv_lib/test/error_test.cpp, 337
csv_lib/test/exception_test.cpp, 339
csv_lib/test/main.cpp, 209
csv_lib/test/read_csv_file_test.cpp, 340
csv_lib/test/s2n_test.cpp, 341
csv_lib/test/sensor_test.cpp, 343
csv_lib/test/to_string_test.cpp, 344
csv_line_test.cpp
TEST, 218
CsvFileInfo
cm::CsvFileInfo, 107
CsvFileKind
cl, 14
CsvLineBuilder
cs::CsvLineBuilder, 110
ctg, 63
aboveThreshold, 64
averageComparisonValueCalculator, 64
halfMaximumComparisonValueCalculator, 65
isRelevant, 66
percentageOf, 67
runAboveThreshold, 68

data_point.cpp
channel, 314
fileName, 314
sensor, 315
time, 315
value, 316
data_point_test.cpp
dp, 330
TEST, 329, 330
data_set_accessor_v
cl, 25
data_set_identifier.cpp
CM_DATA_SET_IDENTIFIER_X, 261
DSI, 261
data_set_identifier.hpp
CM_DATA_SET_IDENTIFIER, 244
CM_DATA_SET_IDENTIFIER_X, 244

data_set_identifier_test.cpp
DSI, 269
TEST, 269
data_set_info.hpp
CS_SPECIALIZE_DATA_SET_INFO, 192
data_set_info_test.cpp
TEST, 218
data_set_test.cpp
EXPECT_LONG_DOUBLE_EQ, 331
TEST, 331–334
DataPoint
cl::DataPoint, 123
dataSet
cs::CsvLineBuilder, 110
dataSetAccessor
cl, 16
DataSetIdentifier
cm, 33
delete_non_bosch_sensors.cpp
CL_SENSOR_X, 356
deleteLowVariance
cs::CsvLineBuilder, 111
cs::LogInfo, 159
deleteNonBoschSensors
fmc, 70
deleteOutOfBoundsValues
fmc, 71
deleteTooClose
cm::Configuration, 87
cm::Configuration::Builder, 77
cs::CsvLineBuilder, 112
cs::LogInfo, 159
deleteTooCloseOptions
cm::Configuration, 87
deleteTooLowVariance
cm::Configuration, 88
cm::Configuration::Builder, 78
deleteTooLowVarianceOptions
cm::Configuration, 88
directory_listing_test.cpp
TEST, 336
directoryListing
cl::fs, 27
DirectoryListingOption
cl::fs, 26
disregardTrueNegativesSorter
cm, 51
distance
cm, 38
distanceScore
cm, 39
distScore
cm::ConfigWithDistanceScore, 99
dos2unix
cl, 17
dp
data_point_test.cpp, 330
DSI
data_set_identifier.cpp, 261
data_set_identifier_test.cpp, 269
manual_segmentation_point.cpp, 265
manual_segmentation_point_test.cpp, 272

Error
cl::Error, 139
error
error_test.cpp, 338
error.cpp
CL_ERROR_KIND_X, 318
error.hpp
CL_ERROR_KIND, 298
CL_ERROR_KIND_X, 299
CL_UNEXPECTED, 299
error_test.cpp
error, 338
TEST, 338
Exception
cl::Exception, 143
exception.hpp
CL_THROW, 300
CL_THROW_FMT, 300
exception_test.cpp
TEST, 339
ExcludeDotAndDotDot
cl::fs, 27
exists
cl::fs::File, 147
cl::fs::Path, 176
EXPECT_LONG_DOUBLE_EQ
above_threshold_test.cpp, 287
data_set_test.cpp, 331
percentage_of_test.cpp, 289
EXPECT_SEGMENTATION_POINTS
segment_test.cpp, 275
Expected
cl, 13
ExtractId
cl, 14
extractId
cl::DataSet, 132

falseNegatives
cm::ConfusionMatrix, 103
falsePositives
cm::ConfusionMatrix, 103
fetch
cm, 40
File
cl::fs::File, 146
file
cl::Error, 139
cl::Exception, 143
cl::Process, 182
fileName
cl::DataPoint, 124
cl::DataSet, 133
cs::LogLine, 164

data_point.cpp, 314
filePath
cs::LogLine, 165
FileStream
cl::fs::FileStream, 153
filter
cs::CsvLineBuilder, 113
FilterKind
cs, 53
filterKind
cm::Configuration, 89
cm::Configuration::Builder, 79
cs::LogInfo, 159
filterKindOptions
cm::Configuration, 89
fix_csv/CMakeLists.txt, 188
fix_csv/include/adjust_hardware_timestamp.hpp, 345
fix_csv/include/convert_to_unix_line_endings.hpp, 346
fix_csv/include/create_backup_file.hpp, 347
fix_csv/include/delete_non_bosch_sensors.hpp, 348
fix_csv/include/delete_out_of_bounds_values.hpp, 349
fix_csv/include/remove_zeros_from_field.hpp, 350
fix_csv/include/restore_from_backup.hpp, 351
fix_csv/include/write_file.hpp, 352
fix_csv/src/adjust_hardware_timestamp.cpp, 353
fix_csv/src/convert_to_unix_line_endings.cpp, 354
fix_csv/src/create_backup_file.cpp, 355
fix_csv/src/delete_non_bosch_sensors.cpp, 355
fix_csv/src/delete_out_of_bounds_values.cpp, 356
fix_csv/src/main.cpp, 210
fix_csv/src/remove_zeros_from_field.cpp, 357
fix_csv/src/restore_from_backup.cpp, 357
fix_csv/src/write_file.cpp, 358
fix_csv/test/adjust_hardware_timestamp_test.cpp, 359
fix_csv/test/CMakeLists.txt, 188
fix_csv/test/main.cpp, 211
fix_csv/test/remove_zeros_from_field_test.cpp, 361
Fixed
cl, 14
fmc, 68
adjustHardwareTimestamp, 69
convertToUnixLineEndings, 69
createBackupFile, 70
deleteNonBoschSensors, 70
deleteOutOfBoundsValues, 71
removeZerosFromField, 71
restoreFromBackup, 72
writeFile, 72
formatError
cl::fs, 28
frame
cm::ManualSegmentationPoint, 170
function
cl::Error, 139
cl::Exception, 144

gyroscopeAverage
cl::DataSet, 133
gyroscopeMaximum
cl::DataSet, 134
gyroscopeThreshold
cl, 25
GyroscopeX
cl, 14
gyroscopeX
cl::DataSet, 135
GyroscopeY
cl, 14
gyroscopeY
cl::DataSet, 135
GyroscopeZ
cl, 14
gyroscopeZ
cl::DataSet, 135

halfMaximumComparisonValueCalculator
ctg, 65
HardwareTimestamp
cl, 14
hardwareTimestamp
cl::DataSet, 136
hardwareTimestamps
cm::CsvFileInfo, 108
hour
cm::ManualSegmentationPoint, 171

importSegmentationPoints
cm::Configuration, 90
Imu
cm, 33
imu
cm::Configuration, 91
cm::Configuration::Builder, 80
imu.cpp
CM_IMU_X, 263
imu.hpp
CM_IMU, 248
CM_IMU_X, 249
imuCount
cm, 51
imuOptions
cm::Configuration, 91
imus
cm, 52
include
CMakeLists.txt, 185–189
incrementFalseNegatives
cm::ConfusionMatrix, 103
incrementFalsePositives
cm::ConfusionMatrix, 104
incrementTrueNegatives
cm::ConfusionMatrix, 104
incrementTruePositives
cm::ConfusionMatrix, 105
interpolated_data_set_paths_test.cpp
TEST, 270
interpolatedDataSetPaths
cm, 41

invalidSensor
cs::LogInfo, 163
cs::LogLine, 167
isAccelerometer
cl, 18
isDirectory
cl::fs::Path, 177
isFile
cl::fs::Path, 178
isGyroscope
cl, 18
isInitialized
cm::Configuration, 92
cs::LogInfo, 160
isOld
cs::CsvLineBuilder, 114
isRelevant
ctg, 66

Kind
cl::Error, 138
kind
cl::Error, 140
cs::CsvLineBuilder, 115

line
cl::Error, 140
cl::Exception, 144
log_files_test.cpp
TEST, 219, 220
log_info_test.cpp
TEST, 222–230
log_line_test.cpp
TEST, 231–233
logFilePath
cs::LogInfo, 160
logFiles
cs, 58
LogInfo
cs::LogInfo, 158
logPath
cs, 63

main
main.cpp, 204, 206, 207, 209–213, 215
main.cpp
main, 204, 206, 207, 209–213, 215
SORT_PRINT, 213
manual_segmentation_point.cpp
DSI, 265
manual_segmentation_point_test.cpp
DSI, 272
TEST, 272–274
ManualSegmentationPoint
cm::ManualSegmentationPoint, 168
matrix
cm::ConfigWithTotalConfusionMatrix, 101
Maxima
cs, 54

message
cl::Error, 140
Minima
cs, 54
minute
cm::ManualSegmentationPoint, 171
Mode
cs, 54
mode.cpp
CS_MODE_X, 216
mode.hpp
CS_MODE, 197
CS_MODE_X, 197
mode_test.cpp
TEST, 234
moveTo
cl::fs::File, 148
MovingAverage
cs, 54

None
cl::fs, 27

oldLogPath
cs, 63
OpenMode
cl::fs::FileStream, 152
operator!=
cm, 42
cm::ManualSegmentationPoint, 173
cs, 59
cs::LogInfo, 162
operator<
cl::fs, 28
cl::fs::Path, 179
cm, 42
cm::Configuration, 96
operator<<
cl, 19–21
cl::DataPoint, 127
cl::Error, 141
cl::fs, 29
cl::fs::Path, 180
cm, 43–45
cm::Configuration, 97
cm::ConfigWithTotalConfusionMatrix, 101
cm::ManualSegmentationPoint, 174
cs, 59, 60
cs::LogInfo, 162
operator()
std::hash<::cl::fs::Path >, 156
std::hash<::cm::Configuration >, 156
operator+=
cm::ConfusionMatrix, 105
operator=
cl::fs::FileStream, 154
cl::Process, 182
operator==
cl::fs, 29

cl::fs::Path, 180
cm, 46
cm::Configuration, 97
cm::ManualSegmentationPoint, 174
cs, 61
cs::LogInfo, 163
orderConfigurationsByQuality
cm, 46

parse
cs::LogLine, 165
parseMode
cs, 61
Path
cl::fs::Path, 176
path
cl::fs::File, 149
percentage_of_test.cpp
EXPECT_LONG_DOUBLE_EQ, 289
TEST, 289
percentageOf
ctg, 67
PL_DEFINE_EXCEPTION_TYPE
cs, 62
PL_NONCOPYABLE
cl::fs::FileStream, 155
cl::Process, 182
Process
cl::Process, 181
python_output.cpp
CM_DEV_NULL, 267
CM_SEGMENTOR, 267
pythonOutput
cm, 47

raise
cl::Error, 141
Raw
cl, 14
Read
cl::fs::FileStream, 153
read_csv_file_test.cpp
TEST, 340, 341
readAll
cl::fs::FileStream, 155
readCsvFile
cl, 21
cm::ManualSegmentationPoint, 172
ReadWrite
cl::fs::FileStream, 153
remove
cl::fs::File, 149
remove_zeros_from_field_test.cpp
TEST, 362–364
removeZerosFromField
fmc, 71
repetitionCount
cs, 62
repetitions
cs::CsvLineBuilder, 116
restoreFromBackup
fmc, 72
rowCount
cl::DataSet, 136
runAboveThreshold
ctg, 68

s2n
cl, 22
s2n_test.cpp
TEST, 342
SamplingRate
cl, 14
second
cm::ManualSegmentationPoint, 173
segment
cm, 48
segment_test.cpp
EXPECT_SEGMENTATION_POINTS, 275
TEST, 275
SegmentationKind
cs, 54
segmentationKind
cm::Configuration, 92
cm::Configuration::Builder, 81
cs::LogInfo, 160
segmentationKindOptions
cm::Configuration, 93
segmentationPointCount
cs::LogLine, 166
segmentationPoints
cs::CsvLineBuilder, 117
Sensor
cl, 14
sensor
cl::DataPoint, 125
cs::CsvLineBuilder, 118
cs::LogInfo, 161
cs::LogLine, 166
data_point.cpp, 315
sensor.cpp
CL_SENSOR_X, 324
sensor.hpp
CL_SENSOR, 310
CL_SENSOR_X, 310, 311
sensor_test.cpp
TEST, 344
sensors
cl, 25
separator.hpp
CL_FS_SEPARATOR, 305
serializeSegmentationPoints
cm::Configuration, 93
set
CMakeLists.txt, 185–189
size
cl::fs::File, 150
size_type

cl::DataSet, 128
skipWindow
cm::Configuration, 94
cm::Configuration::Builder, 82
cs::CsvLineBuilder, 119
cs::LogInfo, 161
skipWindowOptions
cm::Configuration, 94
SORT_PRINT
main.cpp, 213
split_string_test.cpp
TEST, 277
splitString
cm, 50
std::hash< Configuration >
cm::Configuration, 97
std::hash<::cl::fs::Path >, 156
operator(), 156
std::hash<::cm::Configuration >, 156
operator(), 156
str
cl::fs::Path, 178

TEST
above_threshold_test.cpp, 288
adjust_hardware_timestamp_test.cpp, 359–361
channel_test.cpp, 326, 327
column_test.cpp, 328
csv_line_test.cpp, 218
data_point_test.cpp, 329, 330
data_set_identifier_test.cpp, 269
data_set_info_test.cpp, 218
data_set_test.cpp, 331–334
directory_listing_test.cpp, 336
error_test.cpp, 338
exception_test.cpp, 339
interpolated_data_set_paths_test.cpp, 270
log_files_test.cpp, 219, 220
log_info_test.cpp, 222–230
log_line_test.cpp, 231–233
manual_segmentation_point_test.cpp, 272–274
mode_test.cpp, 234
percentage_of_test.cpp, 289
read_csv_file_test.cpp, 340, 341
remove_zeros_from_field_test.cpp, 362–364
s2n_test.cpp, 342
segment_test.cpp, 275
sensor_test.cpp, 344
split_string_test.cpp, 277
to_string_test.cpp, 345
this_type
cl::fs::FileStream, 152
cl::Process, 181
cm::ConfusionMatrix, 102
cs::CsvLineBuilder, 109
threshold
cl, 22
Time
cl, 14
time
cl::DataPoint, 125
cl::DataSet, 137
data_point.cpp, 315
to_string
cl, 23
cl::Error, 141
to_string_test.cpp
TEST, 345
toDataSetIdentifier
cm, 50
totalCount
cm::ConfusionMatrix, 106
Trigger
cl, 14
trigger
cl::DataSet, 137
trueNegatives
cm::ConfusionMatrix, 106
truePositives
cm::ConfusionMatrix, 106

useUnbufferedIo
cl, 24
utf16ToUtf8
cl::fs, 29
utf8ToUtf16
cl::fs, 30

value
cl::DataPoint, 126
data_point.cpp, 316

windowSize
cm::Configuration, 95
cm::Configuration::Builder, 83
cs::CsvLineBuilder, 120
cs::LogInfo, 161
windowSizeOptions
cm::Configuration, 95
Write
cl::fs::FileStream, 153
write
cl::fs::FileStream, 155
writeFile
fmc, 72