

**PixelFormatConverter lib**

**С++ программная библиотека конвертирования**

**изображений в различные форматы пикселов**

Версия библиотеки: **1.0**

Дата релиза библиотеки: **03.01.2020**

Версия документа: **1.0**

[www.zaplatnikov.com](http://www.zaplatnikov.com)

**ОГЛАВЛЕНИЕ**

[1. ВЕРСИИ ДОКУМЕНТА 4](#_Toc28900054)

[2. ВЕРСИИ ПРОГРАММНОЙ БИБЛИОТЕКИ 4](#_Toc28900055)

[3. ОБЗОР 4](#_Toc28900056)

[4. ПОДДЕРЖИВАЕМЫЕ ФОРМАТЫ ПИКСЕЛОВ 4](#_Toc28900057)

[5. КОНВЕРТАЦИЯ МЕЖДУ ФОРМАТАМИ 5](#_Toc28900058)

[5.1. RGBR (RGB24) в BGRB (BGR24) 5](#_Toc28900059)

[5.2. RGBR (RGB24) в Y800 5](#_Toc28900060)

[5.3. RGBR (RGB24) в UYVY 5](#_Toc28900061)

[5.4. RGBR (RGB24) в YUY2 6](#_Toc28900062)

[5.5. RGBR (RGB24) в YUV1 6](#_Toc28900063)

[5.6. RGBR (RGB24) в NV12 6](#_Toc28900064)

[5.7. BGRB (BGR24) в RGBR (RGB24) 6](#_Toc28900065)

[5.8. BGRB (BGR24) в Y800 7](#_Toc28900066)

[5.9. BGRB (BGR24) в UYVY 7](#_Toc28900067)

[5.10. BGRB (BGR24) в YUY2 7](#_Toc28900068)

[5.11. BGRB (BGR24) в YUV1 7](#_Toc28900069)

[5.12. BGRB (BGR24) в NV12 8](#_Toc28900070)

[5.13. Y800 в RGBR (RGB24) 8](#_Toc28900071)

[5.14. Y800 в BGRB (BGR24) 8](#_Toc28900072)

[5.15. Y800 в UYVY 8](#_Toc28900073)

[5.16. Y800 в YUY2 9](#_Toc28900074)

[5.17. Y800 в YUV1 9](#_Toc28900075)

[5.18. Y800 в NV12 9](#_Toc28900076)

[5.19. UYVY в RGBR (RGB24) 10](#_Toc28900077)

[5.20. UYVY в BGRB (BGR24) 10](#_Toc28900078)

[5.21. UYVY в Y800 10](#_Toc28900079)

[5.22. UYVY в YUY2 10](#_Toc28900080)

[5.23. UYVY в YUV1 11](#_Toc28900081)

[5.24. UYVY в NV12 11](#_Toc28900082)

[5.25. YUY2 в RGBR (RGB24) 11](#_Toc28900083)

[5.26. YUY2 в BGRB (BGR24) 12](#_Toc28900084)

[5.27. YUY2 в Y800 12](#_Toc28900085)

[5.28. YUY2 в UYVY 12](#_Toc28900086)

[5.29. YUY2 в YUV1 12](#_Toc28900087)

[5.30. YUY2 в NV12 13](#_Toc28900088)

[5.31. YUV1 в RGBR (RGB24) 13](#_Toc28900089)

[5.32. YUV1 в BGRB (BGR24) 13](#_Toc28900090)

[5.33. YUV1 в Y800 13](#_Toc28900091)

[5.34. YUV1 в UYVY 14](#_Toc28900092)

[5.35. YUV1 в YUY2 14](#_Toc28900093)

[5.36. YUV1 в NV12 14](#_Toc28900094)

[5.37. NV12 в RGBR (RGB24) 15](#_Toc28900095)

[5.38. NV12 в BGRB (BGR24) 15](#_Toc28900096)

[5.39. NV12 в Y800 16](#_Toc28900097)

[5.40. NV12 в UYVY 16](#_Toc28900098)

[5.41. NV12 в YUY2 17](#_Toc28900099)

[5.42. NV12 в YUV1 17](#_Toc28900100)

[6. СТРУКТУРЫ ДАННЫХ 17](#_Toc28900101)

[7. ОПИСАНИЕ КЛАССА PixelFormatConverter 19](#_Toc28900102)

[7.1. Объявление класса PixelFormatConverter 19](#_Toc28900103)

[7.2. Метод Convert(…) 20](#_Toc28900104)

[7.3. Метод GetVersion(…) 20](#_Toc28900105)

[7.4. Метод isFourccCodeValid(…) 20](#_Toc28900106)

[7.5. Метод GetSupportedFourccCodes() 20](#_Toc28900107)

[8. ПРИМЕР ИСПОЛЬЗОВАНИЯ 21](#_Toc28900108)

# ВЕРСИИ ДОКУМЕНТА

Таблица 1 – Версии документа.

|  |  |  |
| --- | --- | --- |
| **Версия** | **Дата релиза** | **Что изменено** |
| 1.0 | 03.01.2020 | Первая версия документа. |

# ВЕРСИИ ПРОГРАММНОЙ БИБЛИОТЕКИ

Таблица 2 – Версии программной библиотеки.

|  |  |  |
| --- | --- | --- |
| **Версия** | **Дата релиза** | **Что изменено** |
| 1.0 | 30.12.2019 | Первая версия программной библиотеки. Реализована конвертация между форматами пикселей RGBR (RGB24), BGRB (BGR24), UYVY, Y800, YUY2, YUV1 и NV12. |

# ОБЗОР

**PixelFormatConverter lib** – это C++ программная библиотека, предназначенная для конвертации изображений в различные форматы пикселов (далее – библиотека). Библиотека имеет простой интерфейс. Библиотека распространяется исходными кодами и совместима с любыми операционными системами, поддерживающими компилятор языка C++ (стандарт C++11). Библиотека включает следующие файлы исходного кода:

* **VideoDataStructures.h** – заголовочный файл, описывающий структуры данных для изображений и кадров видео;
* **PixelFormatConverter.h** – заголовочный файл, содержащий описание единственного программного класса **PixelFormatConverter**;
* **PixelFormatConverter.cpp** – файл исходного кода, содержащий реализацию методов программного класса PixelFormatConverter.

# ПОДДЕРЖИВАЕМЫЕ ФОРМАТЫ ПИКСЕЛОВ

Библиотека поддерживает следующие форматы пикселов: RGBR (RGB24), BGRB (BGR24), UYVY, Y800 (градации серого), YUY2, YUV1 и NV12. Числовые значения форматов пикселов (значение кода FOURCC) определено перечислением **ValidFourccCodes**, объявленном в файле **VideoDataStructures.h**. Библиотека поддерживает конвертацию между указанными форматами. В таблице 3 приведены иллюстрации расположения байт пикселов в различных форматах для изображения размером **4x4** пиксела.

Таблица 3 – Иллюстрации расположения байт данных изображений размером 4x4 пиксела в различных форматах.

|  |  |
| --- | --- |
| Изображение выглядит как небо  Автоматически созданное описание  **Рисунок 1** – RGBR (RGB24) формат пикселов. | Изображение выглядит как небо, стена  Автоматически созданное описание  **Рисунок 2** – BGRB (BGR24) формат пикселов. |
| **Рисунок 3** – UYVY формат пикселов. | Изображение выглядит как кроссворд, текст  Автоматически созданное описание  **Рисунок 4** – YUY2 формат пикселов. |
| Изображение выглядит как текст, кроссворд  Автоматически созданное описание  **Рисунок 5** – Y800 формат пикселов. | **Рисунок 6** – YUV1 формат пикселов. |
| Изображение выглядит как кроссворд, текст  Автоматически созданное описание  **Рисунок 7** – NV12 формат пикселов. |

# КОНВЕРТАЦИЯ МЕЖДУ ФОРМАТАМИ

### 5.1. RGBR (RGB24) в BGRB (BGR24)

|  |
| --- |
|  |
| **Рисунок 8** – Конвертация RGBR (RGB24) в BGRB (BGR24). Только замена байт местами. |

### 5.2. RGBR (RGB24) в Y800

|  |
| --- |
|  |
| **Рисунок 9** – Конвертация RGBR (RGB24) в Y800. |

|  |  |
| --- | --- |
|  | (1) |

### 5.3. RGBR (RGB24) в UYVY

|  |
| --- |
| Изображение выглядит как часы  Автоматически созданное описание |
| **Рисунок 10** – Конвертация RGBR (RGB24) в UYVY. |

|  |  |
| --- | --- |
|  | (2) |

### 5.4. RGBR (RGB24) в YUY2

|  |
| --- |
|  |
| **Рисунок 11** – Конвертация RGBR (RGB24) в YUY2. |

|  |  |
| --- | --- |
|  | (3) |

### 5.5. RGBR (RGB24) в YUV1

|  |
| --- |
| Изображение выглядит как монитор, объект  Автоматически созданное описание |
| **Рисунок 12** – Конвертация RGBR (RGB24) в YUV1. |

|  |  |
| --- | --- |
|  | (4) |

### 5.6. RGBR (RGB24) в NV12

|  |
| --- |
|  |
| **Рисунок 13** – Конвертация RGBR (RGB24) в NV12. |

|  |  |
| --- | --- |
|  | (5) |

### 5.7. BGRB (BGR24) в RGBR (RGB24)

|  |
| --- |
|  |
| **Рисунок 14** – Конвертация BGRB (BGR24) в RGBR (RGB24). Только замена байт местами. |

### 5.8. BGRB (BGR24) в Y800

|  |
| --- |
| Изображение выглядит как оранжевый  Автоматически созданное описание |
| **Рисунок 15** – Конвертация BGRB (BGR24) в Y800. |

|  |  |
| --- | --- |
|  | (6) |

### 5.9. BGRB (BGR24) в UYVY

|  |
| --- |
|  |
| **Рисунок 16** – Конвертация BGRB (BGR24) в UYVY. |

|  |  |
| --- | --- |
|  | (7) |

### 5.10. BGRB (BGR24) в YUY2

|  |
| --- |
|  |
| **Рисунок 17** – Конвертация BGRB (BGR24) в YUY2. |

|  |  |
| --- | --- |
|  | (8) |

### 5.11. BGRB (BGR24) в YUV1

|  |
| --- |
|  |
| **Рисунок 18** – Конвертация BGRB (BGR24) в YUV1. |

|  |  |
| --- | --- |
|  | (9) |

### 5.12. BGRB (BGR24) в NV12

|  |
| --- |
|  |
| **Рисунок 19** – Конвертация BGRB (BGR24) в NV12. |

|  |  |
| --- | --- |
|  | (10) |

### 5.13. Y800 в RGBR (RGB24)

|  |
| --- |
|  |
| **Рисунок 20** – Конвертация Y800 в RGBR (RGB24). |

|  |  |
| --- | --- |
|  | (11) |

### 5.14. Y800 в BGRB (BGR24)

|  |
| --- |
|  |
| **Рисунок 21** – Конвертация Y800 в BGRB (BGR24). |

|  |  |
| --- | --- |
|  | (12) |

### 5.15. Y800 в UYVY

|  |
| --- |
|  |
| **Рисунок 22** – Конвертация Y800 в UYVY. |

|  |  |
| --- | --- |
|  | (13) |

### 5.16. Y800 в YUY2

|  |
| --- |
|  |
| **Рисунок 23** – Конвертация Y800 в YUY2. |

|  |  |
| --- | --- |
|  | (14) |

### 5.17. Y800 в YUV1

|  |
| --- |
|  |
| **Рисунок 24** – Конвертация Y800 в YUV1. |

|  |  |
| --- | --- |
|  | (15) |

### 5.18. Y800 в NV12

|  |
| --- |
| Изображение выглядит как кроссворд, текст  Автоматически созданное описание |
| **Рисунок 25** – Конвертация Y800 в NV12. |

|  |  |
| --- | --- |
|  | (16) |

### 5.19. UYVY в RGBR (RGB24)

|  |
| --- |
|  |
| **Рисунок 26** – Конвертация UYVY в RGBR (RGB24). |

|  |  |
| --- | --- |
|  | (17) |

### 5.20. UYVY в BGRB (BGR24)

|  |
| --- |
|  |
| **Рисунок 27** – Конвертация UYVY в BGRB (BGR24). |

|  |  |
| --- | --- |
|  | (18) |

### 5.21. UYVY в Y800

|  |
| --- |
|  |
| **Рисунок 28** – Конвертация UYVY в Y800. |

|  |  |
| --- | --- |
|  | (19) |

### 5.22. UYVY в YUY2

|  |
| --- |
|  |
| **Рисунок 29** – Конвертация UYVY в YUY2. Только замена байт местами. |

### 5.23. UYVY в YUV1

|  |
| --- |
|  |
| **Рисунок 30** – Конвертация UYVY в YUV1. |

|  |  |
| --- | --- |
|  | (20) |

### 5.24. UYVY в NV12

|  |
| --- |
|  |
| **Рисунок 31** – Конвертация UYVY в NV12. |

|  |  |
| --- | --- |
|  | (21) |

### 5.25. YUY2 в RGBR (RGB24)

|  |
| --- |
|  |
| **Рисунок 32** – Конвертация YUY2 в RGBR (RGB24). |

|  |  |
| --- | --- |
|  | (22) |

### 5.26. YUY2 в BGRB (BGR24)

|  |
| --- |
|  |
| **Рисунок 33** – Конвертация YUY2 в BGRB (BGR24). |

|  |  |
| --- | --- |
|  | (23) |

### 5.27. YUY2 в Y800

|  |
| --- |
|  |
| **Рисунок 34** – Конвертация YUY2 в Y800. |

|  |  |
| --- | --- |
|  | (24) |

### 5.28. YUY2 в UYVY

|  |
| --- |
|  |
| **Рисунок 35** – Конвертация YUY2 в UYVY. Только замена байт местами. |

### 5.29. YUY2 в YUV1

|  |
| --- |
|  |
| **Рисунок 36** – Конвертация YUY2 в YUV1. |

|  |  |
| --- | --- |
|  | (25) |

### 5.30. YUY2 в NV12

|  |
| --- |
|  |
| **Рисунок 37** – Конвертация YUY2 в NV12. |

|  |  |
| --- | --- |
|  | (26) |

### 5.31. YUV1 в RGBR (RGB24)

|  |
| --- |
|  |
| **Рисунок 38** – Конвертация YUV1 в RGBR (RGB24). |

|  |  |
| --- | --- |
|  | (27) |

### 5.32. YUV1 в BGRB (BGR24)

|  |
| --- |
| Изображение выглядит как монитор, экран, телевидение, объект  Автоматически созданное описание |
| **Рисунок 39** – Конвертация YUV1 в RGBR (RGB24). |

|  |  |
| --- | --- |
|  | (28) |

### 5.33. YUV1 в Y800

|  |
| --- |
| Изображение выглядит как часы  Автоматически созданное описание |
| **Рисунок 40** – Конвертация YUV1 в Y800. |

|  |  |
| --- | --- |
|  | (29) |

### 5.34. YUV1 в UYVY

|  |
| --- |
|  |
| **Рисунок 41** – Конвертация YUV1 в UYVY. |

|  |  |
| --- | --- |
|  | (30) |

### 5.35. YUV1 в YUY2

|  |
| --- |
|  |
| **Рисунок 42** – Конвертация YUV1 в YUY2. |

|  |  |
| --- | --- |
|  | (31) |

### 5.36. YUV1 в NV12

|  |
| --- |
|  |
| **Рисунок 43** – Конвертация YUV1 в NV12. |

|  |  |
| --- | --- |
|  | (32) |

### 5.37. NV12 в RGBR (RGB24)

|  |
| --- |
|  |
| **Рисунок 44** – Конвертация NV12 в RGBR (RGB24). |

|  |  |
| --- | --- |
|  | (33) |

### 5.38. NV12 в BGRB (BGR24)

|  |
| --- |
|  |
| **Рисунок 45** – Конвертация NV12 в BGRB (BGR24). |

|  |  |
| --- | --- |
|  | (34) |

### 5.39. NV12 в Y800

|  |
| --- |
| Изображение выглядит как кроссворд, текст  Автоматически созданное описание |
| **Рисунок 46** – Конвертация NV12 в Y800. |

|  |  |
| --- | --- |
|  | (35) |

### 5.40. NV12 в UYVY

|  |
| --- |
|  |
| **Рисунок 47** – Конвертация NV12 в UYVY. |

|  |  |
| --- | --- |
|  | (36) |

### 5.41. NV12 в YUY2

|  |
| --- |
|  |
| **Рисунок 48** – Конвертация NV12 в YUY2. |

|  |  |
| --- | --- |
|  | (37) |

### 5.42. NV12 в YUV1

|  |
| --- |
|  |
| **Рисунок 49** – Конвертация NV12 в YUV1. |

|  |  |
| --- | --- |
|  | (38) |

# СТРУКТУРЫ ДАННЫХ

В качестве входных и выходных данных для программной библиотеки используется класс изображения **Frame**, объявленный в файле **VideoDataStructures.h**. Объявление класса **Frame** приведено ниже.

|  |
| --- |
| class Frame {  public:  uint8\_t\* data;  uint32\_t width;  uint32\_t height;  uint32\_t size;  uint32\_t fourcc;  uint32\_t sourceID;  uint32\_t frameID;  Frame() : data(nullptr), width(0), height(0), size(0), fourcc(0), sourceID(0), frameID(0) { };    Frame(const Frame& src) : data(nullptr), width(0), height(0), size(0), fourcc(0), sourceID(0), frameID(0);  Frame(uint32\_t width, uint32\_t height, uint32\_t fourcc) : data(nullptr), width(0), height(0), size(0), fourcc(0), sourceID(0), frameID(0);  Frame& operator= (const Frame& src);  ~Frame();  void Release();  }; |

Таблица 4 – Описание полей класса **Frame**.

|  |  |
| --- | --- |
| **Поле класса** | **Описание** |
| data | Указатель на массив данных изображения. |
| width | Ширина изображения в пикселах. |
| height | Высота изображения в пикселах. |
| size | Размер данных изображения в байтах. Соответствует размеру буфера данных изображения. |
| fourcc | FOURCC код формата пикселов изображения. В программной библиотеке реализованы следующие форматы пикселов: **RGBR (RGB24)**, **BGRB (BGR24)**, **Y800**, **UYVY**, **YUY2**, **YUV1** и **NV12**. Получить числовое значение кода можно с помощью макроса **MAKE\_FOURCC\_CODE**, объявленного в файле **VideoDataStructures.h**. Также числовые значения разрешенных FOURCC кодов определены в перечислении **ValidFourccCodes**, объявленного в файле **VideoDataStructures.h**:  enum class ValidFourccCodes {  RGB24 = MAKE\_FOURCC\_CODE('R', 'G', 'B', 'R'),  BGR24 = MAKE\_FOURCC\_CODE('B', 'G', 'R', 'B'),  UYVY = MAKE\_FOURCC\_CODE('U', 'Y', 'V', 'Y'),  Y800 = MAKE\_FOURCC\_CODE('Y', '8', '0', '0'),  YUY2 = MAKE\_FOURCC\_CODE('Y', 'U', 'Y', '2'),  YUV1 = MAKE\_FOURCC\_CODE('Y', 'U', 'V', '1'),  NV12 = MAKE\_FOURCC\_CODE('N', 'V', '1', '2'),  JPEG = MAKE\_FOURCC\_CODE('J', 'P', 'E', 'G'),  JPG2 = MAKE\_FOURCC\_CODE('J', 'P', 'G', '2'),  H264 = MAKE\_FOURCC\_CODE('H', '2', '6', '4'),  H265 = MAKE\_FOURCC\_CODE('H', '2', '6', '5')  }; |
| sourceID | Идентификатор источника изображения или видео. Определяется пользователем. |
| frameID | Идентификатор изображения или кадра видео. Может использоваться для нумерации кадров и определяется пользователем. |

Таблица 5 – Описание методов класса **Frame**.

|  |  |
| --- | --- |
| **Метод класса** | **Описание** |
| Frame() | Конструктор класса по умолчанию. |
| ~Frame() | Деструктор класса. |
| Frame(const Frame& src) | Конструктор копирования. |
| Frame(uint32\_t width, uint32\_t height, uint32\_t fourcc) | Конструктор с параметрами:  **width** – ширина изображения в пикселах;  **height** – высота изображения в пикселах;  **fourcc** – FOURCC код формата пикселов.  Конструктор с параметрами выделяем память для данных изображения и заполняет ее нулевыми значениями. |
| Frame& operator= (const Frame& src) | Оператор копирования. |
| void Release() | Метод освобождения памяти. Освобождает память, выделенную для массива данных изображения, и инициализирует все поля значениями по умолчанию. |

# ОПИСАНИЕ КЛАССА PixelFormatConverter

### 7.1. Объявление класса PixelFormatConverter

Класс **PixelFormatConverter** предназначен для конвертации форматов пикселов изображений. Класс объявлен в файле **PixelFormatConvrter.h**. Методы класса не выполняют каких-либо фоновых задач. Расчеты начинаются с вызовом метода **Convert(…)** класса и заканчиваются возвращением управления вызывающему потоку. В качестве входных и выходных данных используются экземпляры класса **Frame**. Ниже приведено объявление класса.

|  |
| --- |
| class PixelFormatConverter {  public:  PixelFormatConverter();  ~PixelFormatConverter();  bool Convert(Frame& src, Frame& dst);  void GetVersion(uint32\_t& major, uint32\_t& minor);  bool isFourccCodeValid(uint32\_t fourcc);  std::vector<uint32\_t> GetSupportedFourccCodes();  }; |

### 7.2. Метод Convert(…)

Метод Conver(…) предназначен для конвертации формата пикселов изображений. Метод конвертирует исходный формат пикселов, указанных во входном изображении (поле fourcc класса Frame) в формат, указанный в объекте класса выходного изображения (поле fourcc класса Frame). Объявление метода:

|  |
| --- |
| bool Convert(Frame& src, Frame& dst); |

*Параметры:*

|  |  |
| --- | --- |
| src | Ссылка на объект класса Frame исходного изображения. **Внимание:** Высота исходного изображения должна быть кратна 2. |
| dst | Ссылка на объект класса Frame результирующего изображения. |

*Возвращаемое значение:*

Метод возвращает TRUE в случае успешной конвертации или возвращает FALSE в следующих случаях:

1. если высота исходного изображения не кратна 2;
2. если FOURCC код исходного изображения (поле fourcc класса Frame) не соответствует списку разрешенных (перечисление ValidFourccCodes в файле VideoDataStructures.h);
3. если FOURCC код результирующего изображения (поле fourcc объекта класса dst) не соответствует списку разрешенных (перечисление ValidFourccCodes в файле VideoDataStructures.h);
4. если исходное изображение src не проинициализировано: размеры изображения равны нулю, нет выделенного массива данных изображения или размер массива данных изображения не соответствует указанному формату пикселей кадров.

### 7.3. Метод GetVersion(…)

Метод GetVersion(…) предназначен для получения числового значения текущей версии программной библиотеки PixelFormatConverter. Объявление метода:

|  |
| --- |
| void GetVersion(uint32\_t& major, uint32\_t& minor); |

*Параметры:*

|  |  |
| --- | --- |
| major | Ссылка на возвращаемое мажорное значение версии библиотеки. |
| minor | Ссылка на возвращаемое минорное значение версии библиотеки. |

### 7.4. Метод isFourccCodeValid(…)

Метод isFourccCodeValid(…) предназначен для проверки соответствия FOURCC кода разрешенному перечню, определенному в перечислении ValidSourccCodes (файл VideoDataStructures.h). Объявление метода:

|  |
| --- |
| bool isFourccCodeValid(uint32\_t fourcc); |

*Параметры:*

|  |  |
| --- | --- |
| fourcc | Код FOURCC для проверки. |

*Возвращаемое значение:*

Метод возвращает TRUE, если переданный в параметрах FOURCC код разрешен. В противном случае метод возвращает FALSE.

### 7.5. Метод GetSupportedFourccCodes()

Метод GetSupportedFourccCodes() предназначен для получения списка разрешенных FOURR кодов. Объявление метода:

|  |
| --- |
| std::vector<uint32\_t> GetSupportedFourccCodes(); |

*Возвращаемое значение:*

Метод возвращает вектор разрешенных FOURCC кодов.

# ПРИМЕР ИСПОЛЬЗОВАНИЯ

Ниже приведен пример конвертации изображения с форматом пикселов YUV1 в BGRB (BGR24). В примере производится сравнение правильности конвертации с конвертацией, реализованной в открытой библиотеке OpenCV.

|  |
| --- |
| const uint32\_t width = 1280;  const uint32\_t height = 1024;  // Create random OpenCV YUV image  cv::Mat yuvImage = cv::Mat(cv::Size(width, height), CV\_8UC3);  for (size\_t i = 0; i < (size\_t)width \* (size\_t)height \* 3; ++i)  yuvImage.data[i] = (uint8\_t)(rand() % 255);  // Create YUV frame  zs::Frame yuvFrame = zs::Frame(width, height, MAKE\_FOURCC\_CODE('Y', 'U', 'V', '1'));  memcpy(yuvFrame.data, yuvImage.data, yuvFrame.size);  // Convert image to RGB with OpenCV  cv::Mat bgrImage;  cv::cvtColor(yuvImage, bgrImage, cv::COLOR\_YUV2BGR);  // Convert frame to RGB  zs::Frame bgrFrame;  bgrFrame.fourcc = MAKE\_FOURCC\_CODE('B', 'G', 'R', 'B');  zs::PixelFormatConverter converter;  if (!converter.Convert(yuvFrame, bgrFrame)) {  Assert::Fail(L"Convert function returned FALSE");  return;  }  // Compare data  uint8\_t val0, val1;  for (size\_t i = 0; i < (size\_t)yuvFrame.size; ++i) {  val0 = bgrFrame.data[i];  val1 = bgrImage.data[i];  if (abs((int)val0 - (int)val1) > 1) {  Assert::Fail(L"Data not equal");  return;  }  } |