DUO Dense3D API

Overview

DUO Dense3D API - Depth Extraction Middleware

Part of the DUO SDK, The DUO Dense3D API provides high level algorithms for depth reconstruction, controls and configuration. This document outlines all functionality of the DUO Dense3D API via the `Dense3D` library which provides a `C` interface paired with `C++` samples.

Here is a overview of the key methods available in this API:

```
// Dense3D initialization
bool Dense3DOpen(Dense3DInstance *dense3D);
bool Dense3DClose(Dense3DInstance dense3D);

// Dense3D error code
bool Dense3DGetErrorCode();

// Dense3D library version
char* Dense3DGetLibVersion();

// Dense3D processing
bool Dense3DGetDepth(Dense3DInstance dense3D, uint8_t *leftImage, uint8_t *rightImage, float *dispar
ityData, PDense3DDepth depthData);

// Dense3D exporting
bool Dense3DSavePLY(Dense3DInstance dense3D, char *plyFile);

// Get Dense3D parameters
bool GetDense3DImageSize(dense3D, uint32_t *w, uint32_t *h);
bool GetDense3DNumDisparities(dense3D, uint32_t *val);
bool GetDense3DSADWindowSize(dense3D, uint32_t *val);
bool GetDense3DP1(dense3D, uint32_t *val);
bool GetDense3DP2(dense3D, uint32_t *val);
bool GetDense3DPreFilterCap(dense3D, uint32_t *val);
bool GetDense3DUniqunessRatio(dense3D, uint32_t *val)
bool GetDense3DSpeckleWindowSize(dense3D, uint32_t *val);
bool GetDense3DSpeckleRange(dense3D, uint32_t *val);

// Set Dense3D parameters
bool SetDense3DLicense(dense3D, char *val);
bool SetDense3DImageSize(dense3D, uint32_t w, uint32_t h);
bool SetDense3DCalibration(dense3D, *intr, *extr);
bool SetDense3DNumDisparities(dense3D, uint32_t val);
bool SetDense3DSADWindowSize(dense3D, uint32_t val);
bool SetDense3DP1(dense3D, uint32_t val);
bool SetDense3DP2(dense3D, uint32_t val);
bool SetDense3DPreFilterCap(dense3D, uint32_t val);
bool SetDense3DUniquenessRatio(dense3D, uint32_t val);
bool SetDense3DSpeckleWindowSize(dense3D, uint32_t val);
bool SetDense3DSpeckleRange(dense3D, uint32_t val);
```

Dense3DInstance

The Dense3D processor instance is a handle filled by the `Dense3DOpen()` and used in subsequent API calls.

```
void *Dense3DInstance
```

Methods

Dense3DOpen

Opens the Dense3D processor and fills the pointer with the Dense3DInstance handle. This function must be called before using any API functions that require Dense3DInstance parameter. (All subsequent calls use Dense3DInstance to access the processor.) The function returns `true` on success.

```
bool Dense3DOpen(Dense3DInstance *dense3D)
```

Dense3DClose

Closes the Dense3D processor. The function returns `true` on success.

```
bool Dense3DClose(Dense3DInstance dense3D)
```

Dense3DGetErrorCode

Returns the current status of the processor with errors from the `Dense3DErrorCode` structure.

```
Dense3DErrorCode *Dense3DGetErrorCode()
```

Dense3DGetLibVersion

Retrieves the library version as a `string`.

```
char *Dense3DGetLibVersion()
```

Dense3DGetDepth

Fills the user allocated Dense3DDepth array pointer with processed depth map. Requires passing of the rectified image pointers. The function returns `true` on success.

```
bool Dense3DGetDepth(Dense3DInstance dense3D, uint8_t *leftImage, uint8_t *rightImage, float *dispar
ityData, PDense3DDepth depthData)
```

Dense3DSavePLY

Returns current depth map as PLY format which can be then saved to disk. The function returns `true` on success.

```
bool Dense3DSavePLY(Dense3DInstance dense3D, char *plyFile)
```

Parameters

Get Parameter Functions

Used to get various parameter values from the library.

GetDense3DImageSize

Returns the width and height of the processed image. The function returns `true` on success.

```
bool GetDense3DImageSize(dense3D, uint32_t *width, uint32_t *height)
```

GetDense3DNumDisparities

Fills the user provided unsigned integer with current Number of Disparities value. The function returns `true` on success.

```
bool GetDense3DNumDisparities(dense3D, uint32_t *val)
```

GetDense3DSADWindowSize

Fills the user provided unsigned integer with current SAD Window value. The function returns `true` on success.

```
bool GetDense3DSADWindowSize(dense3D, uint32_t *val)
```

GetDense3DP1

Fills the user supplied variable with P1 parameter. The function returns `true` on success.

```
bool GetDense3DP1(dense3D, uint32_t *val)
```

GetDense3DP2

Fills the user supplied variable with P2 parameter. The function returns `true` on success.

```
bool GetDense3DP2(dense3D, uint32_t *val)
```

GetDense3DPreFilterCap

Fills the user supplied variable with Prefilter Cap. The function returns `true` on success.

```
bool GetDense3DPreFilterCap(dense3D, uint32_t *val)
```

GetDense3DUniqunessRatio

Fills the user provided unsigned integer with current Uniqueness Ratio value. The function returns `true` on success.

```
bool GetDense3DUniqunessRatio(dense3D, uint32_t *val)
```

GetDense3DSpeckleWindowSize

Fills the user supplied variable with Speckle Window Size. The function returns `true` on success.

```
bool GetDense3DSpeckleWindowSize(dense3D, uint32_t *val)
```

GetDense3DSpeckleRange

Fills the user supplied variable with current Speckle Range. The function returns `true` on success.

```
bool GetDense3DSpeckleRange(dense3D, uint32_t *val)
```

Set Parameter Functions

Used to set various parameter values from the library.

SetDense3DLicense

Set your license to after initialization of the processor. The function returns `true` on success.

```
bool SetDense3DLicense(dense3D, char *val)
```

SetDense3DImageSize

Sets the processed image size. The function returns `true` on success.

```
bool SetDense3DImageSize(dense3D, uint32_t *width, uint32_t *height)
```

SetDense3DCalibration

Sets the active calibration data, accepting file paths for the intrinsic and extrinsic data. The function returns `true` on success.

```
bool SetDense3DCalibration(dense3D, void *intr_file, void *extr_file)
```

SetDense3DNumDisparities

Sets the number of disparity steps (range [2, 16]) The function returns `true` on success.

```
bool SetDense3DNumDisparities(dense3D, uint32_t *val)
```

SetDense3DSADWindowSize

Set the SAD Window Size (range [2, 10]) The function returns `true` on success.

```
bool SetDense3DSADWindowSize(dense3D, uint32_t *val)
```

SetDense3DP1

Set Uniqueness P1 Parameter (range [0, 1600]) The function returns `true` on success.

```
bool SetDense3DP1(dense3D, uint32_t val)
```

SetDense3DP2

Set Uniqueness P2 Parameter (range [0, 3200]) The function returns `true` on success.

```
bool SetDense3DP2(dense3D, uint32_t val)
```

SetDense3DPreFilterCap

Set Pre-Filter (range [0, 256]) The function returns `true` on success.

```
bool SetDense3DPreFilterCap(dense3D, uint32_t val)
```

SetDense3DUniquenessRatio

Set Uniqueness Ratio Post-Filter (range [0, 100]) The function returns `true` on success.

```
bool SetDense3DUniquenessRatio(dense3D, uint32_t *val)
```

SetDense3DSpeckleWindowSize

Set Speckle Window Size (range [0, 256]) The function returns `true` on success.

```
bool SetDense3DSpeckleWindowSize(dense3D, uint32_t val)
```

SetDense3DSpeckleRange

Set Speckle Range (range [0, 16]) The function returns `true` on success.

```
bool SetDense3DSpeckleRange(dense3D, uint32_t val)
```

Parameter Values/Structures

PDense3DDepth

Contains coordinates as floats for storage of depth data (x,y,z).

```
typedef struct
{
    float x;
    float y;
    float z;
}Dense3DDepth, *PDense3DDepth;
```

Dense3DParameter

Contains processor parameters for retrieving and updated internal data.

```
enum Dense3DParameter
{
    // Dense3D parameters
    DENSE3D_LICENSE,                    // Set only: (char*) license key
    DENSE3D_IMAGE_SIZE,                 // Set/Get: (uint32_t, uint32_t) width, height
    DENSE3D_CALIBRATION,                // Set only: (void*) intrinsics, (void*) extrinsics
    DENSE3D_NUM_DISPARITIES,            // Set/Get: (uint32_t) [2, 16]
    DENSE3D_SAD_WINDOW_SIZE,            // Set/Get: (uint32_t) [2, 10]
    DENSE3D_P1,                         // Set/Get: (uint32_t) [0, 1600]
    DENSE3D_P2,                         // Set/Get: (uint32_t) [0, 3200]
    DENSE3D_PREFILTER_CAP,              // Set/Get: (uint32_t) [0, 256]
    DENSED3_UNIQUENESS_RATIO,           // Set/Get: (uint32_t) [0, 100]
    DENSED3_SPECKLE_WINDOW_SIZE,        // Set/Get: (uint32_t) [0, 256]
    DENSED3_SPECKLE_RANGE,              // Set/Get: (uint32_t) [0, 16]
};
```

Dense3DErrorCode

Contains standard status error codes for debugging and troubleshooting.

```
// Dense3D error codes
enum Dense3DErrorCode
{
    DENSE3D_NO_ERROR,
    DENSE3D_INVALID_DENSE3D_INSTANCE,
    DENSE3D_ERROR_CREATING_DENSE3D_INSTANCE,
    DENSE3D_INVALID_LICENSE,
    DENSE3D_INVALID_PARAMETER,
    DENSE3D_INVALD_IMAGE_POINTER,
    DENSE3D_INVALD_DEPTH_DATA_POINTER,
    DENSE3D_INVALID_IMAGE_SIZE,
    DENSE3D_INVALD_PLY_FILE_NAME,
    DENSE3D_ERROR_EXPORTING_PLY_FILE
};
```

Sample Usage

You can easily integrate Dense3D into application or system by using a portable `C` interface that allows for full access and control to the processor.

View the sample code for creating your first application with Dense3D.

Review the samples provided with the SDK:

- Using the Dense3D Middleware

Resources

Tips

- Generate your calibration files using the DUO Dashboard Calibration App.
- Most API calls return a true or false if the action was successful.
- Make sure you USB Hub meets requirements stated in the requirements.
- Always make sure you have the latest DUO SDK, Dashboard and Driver.

Related

Relevant articles and links:

- DUO API
- DUO SDK
- DUO Developers
- DUO Devices
- DUO Downloads