Light Sequence Switch

# LSS-2404

## Photometric Stereo / Shape from Shading SDK Manual



www.ccsamerica.com

www.computationalimaging.com

Version 1.0

## Overview

Shape from Shading is one technique of computational imaging. Computational imaging refers to capturing multiple images and using image processing to generate a better output image which has features or quality no single shot image can provide.

Shape from Shading is a technique used to separate the shape of an object from its 2D texture or surface coloring. Typically, it's used to highlight 3D surface structure information in one image, known as the *shape image*, and remove glare from highly reflective parts, known as the *texture image*. The technique works by using a segmented ring light or 4 independent bar lights and combining multiple images into one. Four images are taken as the light rotates around the part in a counterclockwise direction, illuminating the part in a different direction for each image capture. By combining the resulting shadow images in a process known as shape from shading, the 3D surface structure of a part can be accentuated.

CCS_CI_SDK is a .NET DLL which can generate a shape image or texture image from a set of input images, using a shape from shading algorithm. It requires a licensed CCS LSS-2404 sequencing controller for authorization. LSS-2404 is a dedicated controller which was developed for computational imaging. See detailed information at the following website http://www.computationalimaging.com/ or download LSS-2404 sales literature at the following location http://bit.ly/CCSLIGHTS.

The CCS_CI_SDK has 2 classes:

- CI_SFS class for shape from shading algorithm to generate shape or texture image.
- CI_MCOR class to calculate the part motion movement between each image to feed back to CI_SFS class for motion correction.

For stationary applications, only CI_SFS class is needed. If the part is moving during the image acquisition, both CI_SFS and CI_MCOR classes need to be used.

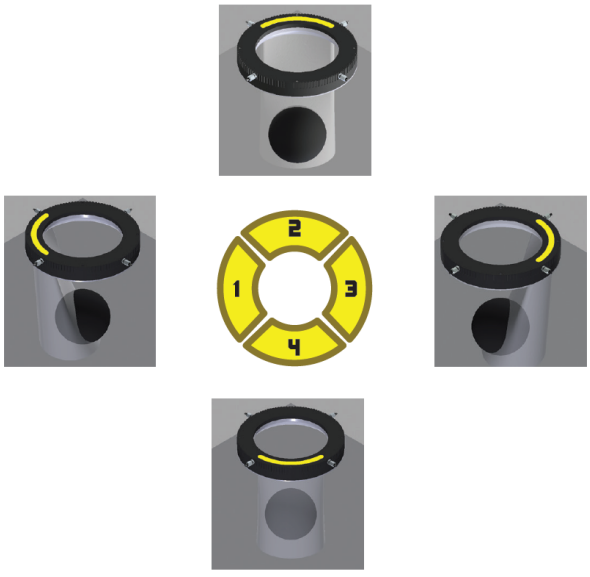
## System requirements

PC: Windows 10, 64 bit

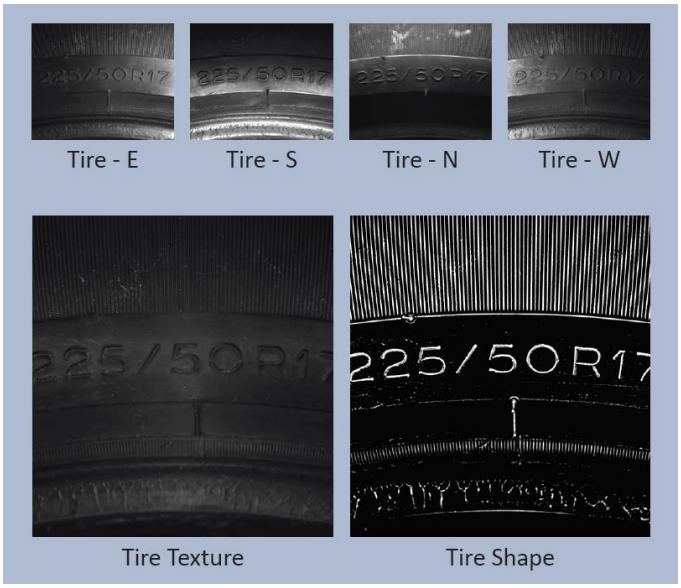
.NET framework: 4.6.2 and above

Microsoft Visual Studio 2015 (later versions may work but are not officially supported)

## Principle of Shape from Shading

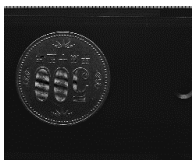



## Application Example: Tire Sidewall



The chart below shows an example of how the images look as the light moves around the part from North to East and includes a search image for motion correction. See the Motion Correction section below for a description of moving parts.

The terms used in the instructions refer to the LSS images based on the direction of the light when the image is taken.

| | | |
|---|---|---|
|  | Input Images | North Image – light coming from the North direction (top of the part)<br>This is also considered the Template Image for motion correction |
|  | | West Image – Light coming from the West direction (left of the part) |
|  | | South Image – Light coming from the South direction (bottom of the part) |
|  | | East Image – Light coming from the East direction (right of the part) |
|  | Search Image | Search Image – used to find the previously defined pattern to determine how the part moved in the field of view |

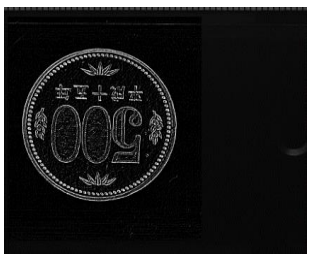
## Shape Image

The algorithm looks for the shadows that are created and the edges that are highlighted as the light shines from each direction. The shadows and edges from each image are combined into one image that represents the height variation in a part. This is very helpful to detect scratches, dents, missing components, etc.

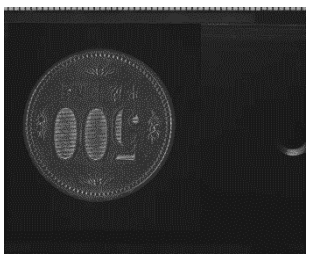
Note: This image will highlight a height change but will not quantify the change. In other words, no precise 3D measurements can be determined from the Shape image.

## Texture Image

The algorithm looks for the area of the images that do not have direct lighting. It then combines all the areas of the image that look the same, meaning they do not have glare, into one image with no glare. This is useful for inspections of parts that are highly reflective or inspecting parts through plastic or clear material.



*Shape Image*



*Texture Image*

## Parameters:

There are 3 parameters which can be adjusted to optimize the shape image.

1. *Kernel:* controls the amount of noise but will also affect the sharpness of the edges
    a. Range: 0-10. Smaller values have little noise reduction along with sharper edges. Larger values result in more noise reduction but blur the edges more.

    Hint: Surface finishes with more texture may require higher values

2. *Contrast:* adjusts the contrast between the white and black
    a. Range: 1-30. Larger values result in higher contrast

    Hint: If the raw images are low in contrast, try larger contrast values

3. Brightness: adjusts the background color
    a. Range: -150 to 150. 0 is a black background and 150 is a white background. The middle of that range is grey. 0 to -150 will remove smaller features while the background remains black.

    Hint: If the part has only indented features, they will appear black so start with higher values. If the part has only raised features, they will appear white so start around 0.

## Motion Correction

If the part is in motion during the inspection the motion correction class is required. This API accounts for the part being in different positions as each image is captured. It requires an extra image after the 4 directional images are taken, known as the *search image*. The search image will have the same lighting direction as the first image.

In the API, a pattern is defined on the first image. Then a search region, or ROI, is defined on the search image. The search region is drawn where the defined pattern will fall in the search image. Once it finds the pattern, the algorithm determines how the part moved during the image capture sequence, so it can identify and align the correct pixels. The definitions of each image and region of interest are described below.

*Template ROI*- defines the pattern that will be located in the Search Image to determine how far the part has moved in the FOV

*Search ROI* - area within the search image where we expect to find the pattern at the end of the capture sequence

*Merged Image* – The output image of which a region will have SFS processing performed. Merged Image can be either shape image or texture image.

*Merge ROI* – defines the portion of the output image that will have the SFS function. This may encompass the entire part or just a region of interest for SFS.

If your part moves be sure to set your camera's field of view large enough that the part stays within the field of view during the entire capture sequence. The pattern search does not have the capability to rotate or scale the pattern, so the part must move straight at constant speed and not vary in size (e.g. not down an inclined plane).

*Tip: be sure to consider the resolution necessary to accommodate the extra image needed for motion correction.*

### Parameters:

There are 3 parameters which can be adjusted to optimize the shape image.

1. Threshold
    - Adjust the acceptance level low enough where it finds the pattern, but not too low where it finds the incorrect pattern (generally 30 is a good starting point)

2. featureContrast
    - Increasing the feature size will select the most obvious edges.

3. noiseRemoval
    - Increasing the noise level will remove the smaller or less obvious edges.

    If Feature or Noise is set to -1, the algorithm will automatically set the optimal parameters. -1 should be sufficient to determine the best pattern, therefore only adjust these parameters if too many edges are found.

    The feature value must always be higher than the noise value.

**API Description:**

| Name | Overload | Description |
|---|---|---|
| CI_SFS Class | | This class inputs 4 images and outputs shape image or texture image based on the shape from shading algorithm |
| Member | | |
| public CIErrorCode SFSError | | |
| public bool AuthorizationFlag | | |
| public bool SetUpFlag | | |
| Function | | |
| public bool AuthorizationCheck(string ipAddress) | | |
| public void ResetFlag() | | |
| public bool InputImageNorth | public bool InputImageNorth(int imgWidth, int imgHeight, IntPtr imgBuffer)<br><br>public bool InputImageNorth(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer)<br><br>public bool InputImageNorth(string file_name) | |
| public bool InputImageWest | public bool InputImageWest(int imgWidth, int imgHeight, IntPtr imgBuffer)<br><br>public bool InputImageWest(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer)<br><br>public bool InputImageWest(string file_name) | |
| public bool InputImageSouth | public bool InputImageSouth(int imgWidth, int imgHeight, IntPtr imgBuffer)<br><br>public bool InputImageSouth(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer)<br><br>public bool InputImageSouth(string file_name) | |

| | | |
|---|---|---|
| public bool InputImageEast | public bool InputImageEast(int imgWidth, int imgHeight, IntPtr imgBuffer)<br><br>public bool InputImageEast(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer)<br><br>public bool InputImageEast(string file_name) | |
| public bool MergeShapeSFS | public bool MergeShapeSFS(int kernel, double contrast, double brightness, ref IntPtr mergeImageBuffer)<br><br>public bool MergeShapeSFS(int kernel, double contrast, double brightness, int imgStep, ref IntPtr mergeImageBuffer)<br><br>public bool MergeShapeSFS(int kernel, double contrast, double brightness, int roiX, int roiY, int roiWidth, int roiHeight, int moveX, int moveY, ref IntPtr mergeImageBuffer)<br><br>public bool MergeShapeSFS(int kernel, double contrast, double brightness, int roiX, int roiY, int roiWidth, int roiHeight, int moveX, int moveY, int imgStep, ref IntPtr mergeImageBuffer) | |
| public bool MergeTextureSFS | public bool MergeTextureSFS(ref IntPtr mergeImageBuffer)<br><br>public bool MergeTextureSFS(int imgStep, ref IntPtr mergeImageBuffer)<br><br>public bool MergeTextureSFS(int roiX, int roiY, int roiWidth, int roiHeight, int moveX, int moveY, ref IntPtr mergeImageBuffer)<br><br>public bool MergeTextureSFS(int roiX, int roiY, int roiWidth, int roiHeight, int moveX, int moveY, int imgStep, ref IntPtr mergeImageBuffer) | |
| | **Application Example** | |
| CI_MCOR Class | | |
| Member | | |
| public CIErrorCode MCORError | | |
| public bool AuthorizationFlag | | |

| | | |
|---|---|---|
| public bool SetUpFlag | | |
| Function | | |
| public bool AuthorizationCheck(string ipAddress) | | |
| public void ResetFlag() | | |
| public bool InputImageTemp | public bool InputImageTemp(int imgWidth, int imgHeight, IntPtr imgBuffer, int roiX, int roiY, int roiWidth, int roiHeight, int noiseRemoval, int featureContrast, bool flagShowOutput, ref IntPtr outputTempImage)<br><br>public bool InputImageTemp(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer, int roiX, int roiY, int roiWidth, int roiHeight, int noiseRemoval, int featureContrast, bool flagShowOutput, int outImgStep, ref IntPtr outputTempImage)<br><br>public bool InputImageTemp(int imgWidth, int imgHeight, IntPtr imgBuffer, int roiX, int roiY, int roiWidth, int roiHeight, int noiseRemoval, int featureContrast)<br><br>public bool InputImageTemp(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer, int roiX, int roiY, int roiWidth, int roiHeight, int noiseRemoval, int featureContrast) | |
| public bool InputImageTempEdge | public bool InputImageTempEdge(int imgWidth, int imgHeight, IntPtr imgBuffer, int roiX, int roiY, int roiWidth, int roiHeight, int noiseRemoval, int featureContrast, bool flagShowOutput, ref IntPtr outputTempImage)<br><br>public bool InputImageTempEdge(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer, int roiX, int roiY, int roiWidth, int roiHeight, int noiseRemoval, int featureContrast, bool flagShowOutput, int outImgStep, ref IntPtr outputTempImage) | |
| public bool MatchImage | public bool MatchImage(int imgWidth, int imgHeight, IntPtr imgBuffer, int roiX, int roiY, int roiWidth, int roiHeight, int threshold, bool flagShowOutput, ref IntPtr outputMatchImage, ref int matchScore, ref int moveDistanceX, ref int moveDistanceY)<br><br>public bool MatchImage(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer, int | |

| | | |
|---|---|---|
| | roiX, int roiY, int roiWidth, int roiHeight, int threshold, bool flagShowOutput, int outImgStep, ref IntPtr outputMatchImage, ref int matchScore, ref int moveDistanceX, ref int moveDistanceY)<br><br>public bool MatchImage(int imgWidth, int imgHeight, IntPtr imgBuffer, int roiX, int roiY, int roiWidth, int roiHeight, int threshold, ref int matchScore, ref int moveDistanceX, ref int moveDistanceY)<br>public bool MatchImage(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer, int roiX, int roiY, int roiWidth, int roiHeight, int threshold, ref int matchScore, ref int moveDistanceX, ref int moveDistanceY) | |
| public bool MatchImageEdge | public bool MatchImageEdge(int imgWidth, int imgHeight, IntPtr imgBuffer, int roiX, int roiY, int roiWidth, int roiHeight, int threshold, bool flagShowOutput, ref IntPtr outputMatchImage, ref int matchScore, ref int moveDistanceX, ref int moveDistanceY)<br><br>public bool MatchImageEdge(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer, int roiX, int roiY, int roiWidth, int roiHeight, int threshold, bool flagShowOutput, int outImgStep, ref IntPtr outputMatchImage, ref int matchScore, ref int moveDistanceX, ref int moveDistanceY) | |
| | **Application Example** | |

# SHAPE FROM SHADING CORRECTION CLASS

```
namespace CCS_CI_SDK

public class CI_SFS
This class inputs 4 images captured with directional lighting and outputs shape image or texture
image based on the shape from shading algorithm.
```

**Member:**

**public CIErrorCode SFSError**

Error handle. Error state code.

enum CIErrorCode

| Value | Meaning |
|---|---|
| 0 | "Unable to open the connect to LSS, check IP address of LSS controller and PC; check whether firewall settings or security software are blocking LSS" |
| 1 | "LSS controller does not have valid license key" |
| 2 | "Invalid pointer. Ensure pointer points to valid data." |
| 3 | "One or more image parameters are invalid. Image width and height must be greater than 0, step size must be greater than width." |
| 4 | "One or more function parameters are invalid. Parameters vary by function, check documentation for parameter details and valid values." |
| 5 | "Unable to open image file" |
| 6 | "License initialization warning. Must authorize LSS license before any CI function call." |
| 7 | "Input image error. One or more input images are missing, invalid or not same size. Check for proper image size and data format." |
| 8 | "One or more ROI parameters are invalid. ROI X/Y must be greater than or equal to 0, ROI width/height must be greater than 0. ROI must be fully contained within the image." |
| 9 | "Unable to complete motion correction. Merge ROI outside FOV after motion compensation." |
| 10 | "Search ROI size invalid. Make sure search ROI is larger than template ROI." |
| 11 | "Unable to find motion correction pattern. Pattern match score is below the threshold." |
| 99 | No Error |

**public bool AuthorizationFlag**

Authorization state code

| Value | Meaning |
|---|---|
| True | Authorization finished |
| False | Authorization not done yet |

**public bool SetUpFlag**

Flag whether all input images are in place

| Value | Meaning |
|---|---|
| True | All input images are in place |
| False | Some input images are missing or have error |

**Function:**

**public bool AuthorizationCheck(string ipAddress)**

Connect to LSS controller and unlock the functions.

**Parameter:**

| | |
|---|---|
| ipAddress | IP Address of LSS controller<br>* remove the 0 in front of number. For example, 192.168.0.2, not 192.168.000.002. |

**Return Value:**

| | |
|---|---|
| True | Success |
| False | Fail |

```
public void ResetFlag()
```
Reset the SetUpFlag (flag whether all input images are in place) to false. Usually called before InputImage functions below.

```
public bool InputImageNorth(int imgWidth, int imgHeight, IntPtr imgBuffer)
```
Fill the image buffer using directional light from North and prepare image for merging. Only supports 8-bit monochrome image data.

**Parameters:**

| | |
|---|---|
| imgWidth | width of image |
| imgHeight | height of image |
| imgBuffer | Pointer to image data |

```
public bool InputImageNorth(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer)
```
Fill the image buffer using directional light from North and prepare image for merging. Only supports 8-bit monochrome image data.

**Parameters:**

| | |
|---|---|
| imgWidth | width of image |
| imgHeight | height of image |
| imgStep | stride of image |
| imgBuffer | Pointer to image data |

```
public bool InputImageNorth(string file_name)
```
Fill the image path, load image from an image file using directional light from North and prepare image for merging. Supports .bmp, .jpg, .png, .tif image files in 8-bit, 16-bit and 32-bit image formats. This function will convert all images to 8-bit grayscale format and process.

**Parameter:**

| | |
|---|---|
| file_name | Path of image file |

```
public bool InputImageWest(int imgWidth, int imgHeight, IntPtr imgBuffer)
```
Fill the image buffer using directional light from West and prepare image for merging. Only supports 8-bit monochrome image data.

**Parameters:**

| | |
|---|---|
| imgWidth | width of image |
| imgHeight | height of image |
| imgBuffer | Pointer to image data |

```
public bool InputImageWest(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer)
```
Fill the image buffer using directional light from West and prepare image for merging. Only supports 8-bit monochrome image data.

**Parameters:**

| | |
|---|---|
| imgWidth | width of image |
| imgHeight | height of image |
| imgStep | stride of image |
| imgBuffer | pointer to image data |

```
public bool InputImageWest(string file_name)
```

public bool InputImageNorth(string file_name)
Fill the image path, load image from an image file using directional light from West and prepare image for merging. Supports .bmp, .jpg, .png, .tif image files in 8-bit, 16-bit and 32-bit image formats. This function will convert all images to 8-bit grayscale format and process.

**Parameter:**
file_name Path of image file

**public bool InputImageSouth(int imgWidth, int imgHeight, IntPtr imgBuffer)**
Fill the image buffer using directional light from South and prepare image for merging. Only supports 8-bit monochrome image data.

**Parameters:**
imgWidth width of image
imgHeight height of image
imgBuffer Pointer to image data

**public bool InputImageSouth(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer)**
Fill the image buffer using directional light from South and prepare image for merging. Only supports 8-bit monochrome image data.

**Parameters:**
imgWidth width of image
imgHeight height of image
imgStep stride of image
imgBuffer Pointer to image data

**public bool InputImageSouth(string file_name)**
Fill the image path, load image from an image file using directional light from South and prepare image for merging. Supports .bmp, .jpg, .png, .tif image files in 8-bit, 16-bit and 32-bit image formats. This function will convert all images to 8-bit grayscale format and process.

**Parameter:**
file_name Path of image file

**public bool InputImageEast(int imgWidth, int imgHeight, IntPtr imgBuffer)**
Fill the image buffer using directional light from East and prepare image for merging. Only supports 8-bit monochrome image data.

**Parameters:**
imgWidth width of image
imgHeight height of image
imgBuffer Pointer to image data

**public bool InputImageEast(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer)**
Fill the image buffer using directional light from East and prepare image for merging. Only supports 8-bit monochrome image data.

**Parameters:**
imgWidth width of image
imgHeight height of image
imgStep stride of image
imgBuffer Pointer to image data

**public bool InputImageEast(string file_name)**

Fill the image path, load image from an image file using directional light from East and prepare image for merging. Supports .bmp, .jpg, .png, .tif image files in 8-bit, 16-bit and 32-bit image formats. This function will convert all images to 8-bit grayscale format and process.

**Parameter:**

| | |
|---|---|
| file_name | Path of image file |

```
public bool MergeShapeSFS(int kernel, double contrast, double brightness, ref IntPtr mergeImageBuffer)
```

Merge four input images and output the result shape image.

**Parameters:**

| | |
|---|---|
| kernel | feature size (0 to 10) |
| contrast | contrast (1 to 30) |
| brightness | brightness (-150 to 150)<br>Positive values for sunken feature, negative values for raised feature |
| mergeImageBuffer | pointer to image data of shape image |

```
public bool MergeShapeSFS(int kernel, double contrast, double brightness, int imgStep, ref IntPtr
mergeImageBuffer)
```

Merge four input images and output the result shape image in specified stride format

**Parameters:**

| | |
|---|---|
| kernel | feature size (0 to 10) |
| contrast | contrast (1 to 30) |
| brightness | brightness (-150 to 150)<br>Positive values for sunken feature, negative values for raised feature |
| imgStep | stride of shape image |
| mergeImageBuffer | Pointer to image data of shape image |

```
public bool MergeShapeSFS(int kernel, double contrast, double brightness, int roiX, int roiY, int
roiWidth, int roiHeight, int moveX, int moveY, ref IntPtr mergeImageBuffer)
```

Merge four input images and output the result shape image with motion correction

**Parameters:**

| | |
|---|---|
| kernel | feature size (0 to 10) |
| contrast | contrast (1 to 30) |
| brightness | brightness (-150 to 150)<br>Positive values for sunken feature, negative values for raised feature |
| roiX | x coordinate of ROI based on North image |
| roiY | y coordinate of ROI based on North image |
| roiWidth | width of ROI based on North image |
| roiHeight | height of ROI based on North image |
| moveX | move distance (pixels) between each input image (X axis) |
| moveY | move distance (pixels) between each input image (Y axis) |
| mergeImageBuffer | Pointer to image data of shape image |

```
public bool MergeShapeSFS(int kernel, double contrast, double brightness, int roiX, int roiY, int
roiWidth, int roiHeight, int moveX, int moveY, int imgStep, ref IntPtr mergeImageBuffer)
```

Merge four input images and output the result shape image in specified stride format with motion correction

**Parameters:**

| | |
|---|---|
| kernel | feature size (0 to 10) |
| contrast | contrast (1 to 30) |
| brightness | brightness (-150 to 150)<br>Positive values for sunken feature, negative values for raised feature |
| roiX | x coordinate of ROI based on North image |

| | |
|---|---|
| roiY | y coordinate of ROI based on North image |
| roiWidth | width of ROI base on North image |
| roiHeight | height of ROI base on North image |
| moveX | move distance (pixels) between each input image (X axis) |
| moveY | move distance (pixels) between each input image (Y axis) |
| imgStep | stride of shape image |
| mergeImageBuffer | Pointer to image data of shape image |

```
public bool MergeTextureSFS(ref IntPtr mergeImageBuffer)
```

Merge four input images and output the result texture image

**Parameter:**

| | |
|---|---|
| mergeImageBuffer | Pointer to image data of texture image |

```
public bool MergeTextureSFS(int imgStep, ref IntPtr mergeImageBuffer)
```

Merge four input images and output the result texture image in specified stride format

**Parameters:**

| | |
|---|---|
| imgStep | stride of texture image |
| mergeImageBuffer | Pointer to image data of texture image |

```
public bool MergeTextureSFS(int roiX, int roiY, int roiWidth, int roiHeight, int moveX, int moveY,
ref IntPtr mergeImageBuffer)
```

Merge four input images and output the result texture image with motion correction

**Parameters:**

| | |
|---|---|
| roiX | x coordinate of ROI based on North image |
| roiY | y coordinate of ROI based on North image |
| roiWidth | width of ROI based on North image |
| roiHeight | height of ROI based on North image |
| moveX | move distance (pixels) between each input image (X axis) |
| moveY | move distance (pixels) between each input image (Y axis) |
| mergeImageBuffer | Pointer to image data of texture image |

```
public bool MergeTextureSFS(int roiX, int roiY, int roiWidth, int roiHeight, int moveX, int moveY,
int imgStep, ref IntPtr mergeImageBuffer)
```

Merge four input images and output the result texture image in specified stride format with motion correction

**Parameters:**

| | |
|---|---|
| roiX | x coordinate of ROI based on North image |
| roiY | y coordinate of ROI based on North image |
| roiWidth | width of ROI based on North image |
| roiHeight | height of ROI based on North image |
| moveX | move distance (pixels) between each input image (X axis) |
| moveY | move distance (pixels) between each input image (Y axis) |
| imgStep | stride of texture image |
| mergeImageBuffer | Pointer to image data of texture image |

**Application Example in C#**

```
using CCS_CI_SDK;

CI_SFS mySFS;

int imgWidth = 640;
int imgHeight = 480;
```

```
int imgStep = 640;

bool shapeSuccess = false;
bool textureSuccess = false;

int myKernal = 1;
int myBrightness = 3;
int myContrast = 0;

IntPtr ptrInputImageNorth = ptrOriginal1;
IntPtr ptrInputImageWest = ptrOriginal2;
IntPtr ptrInputImageSouth = ptrOriginal3;
IntPtr ptrInputImageEast = ptrOriginal4;
IntPtr ptrMergeImageShape = new IntPtr();
IntPtr ptrMergeImageTexture = new IntPtr();
// ptrOriginal1 is pointer to 640x480 image data of North Image.
// ptrOriginal2 is pointer to 640x480 image data of West Image.
// ptrOriginal3 is pointer to 640x480 image data of South Image.
// ptrOriginal4 is pointer to 640x480 image data of East Image.

mySFS = new CI_SFS();

//Authorization - the default IP address of LSS is 192.168.0.10
mySFS.AuthorizationCheck("192.168.0.10");

//Run SFS class to create the final shape image and texture image.
mySFS.ResetFlag();
mySFS.InputImageNorth(imgWidth, imgHeight, imgStep, ptrInputImageNorth);
mySFS.InputImageWest(imgWidth, imgHeight, imgStep, ptrInputImageWest);
mySFS.InputImageSouth(imgWidth, imgHeight, imgStep, ptrInputImageSouth);
mySFS.InputImageEast(imgWidth, imgHeight, imgStep, ptrInputImageEast);

if (mySFS.SetUpFlag)
{
       shapeSuccess = mySFS.MergeShapeSFS(myKernal, myBrightness, myContrast, ref
       ptrMergeImageShape);
       textureSuccess = mySFS.MergeTextureSFS(ref ptrMergeImageTexture);
}

//Check the error code of mySFS.SFSError here
```

## MOTION CORRECTION CLASS

```
public class CI_MCOR
```

This class requires 2 input images with certain position change. It will calculate the movement distance. This movement distance can be used as CI_SFS motion correction input.

Member:

```
public CIErrorCode MCORError
```

Error handle. Error state code.

enum CIErrorCode

| Value | Meaning |
|---|---|
| 12 | "Unable to open the connect to LSS, check IP address of LSS controller and PC; check whether firewall settings or security software are blocking LSS" |
| 13 | "LSS controller does not have valid license key" |
| 14 | "Invalid pointer. Ensure pointer points to valid data." |
| 15 | "One or more image parameters are invalid. Image width and height must be greater than 0, step size must be greater than width." |
| 16 | "One or more function parameters are invalid. Parameters vary by function, check documentation for parameter details and valid values." |
| 17 | "Unable to open image file" |
| 18 | "License initialization warning. Must authorize LSS license before any CI function call." |
| 19 | "Input image error. One or more input images are missing, invalid or not same size. Check for proper image size and data format." |
| 20 | "One or more ROI parameters are invalid. ROI X/Y must be greater than or equal to 0, ROI width/height must be greater than 0. ROI must be fully contained within the image." |
| 21 | "Unable to complete motion correction. Merge ROI outside FOV after motion compensation." |
| 22 | "Search ROI size invalid. Make sure search ROI is larger than template ROI." |
| 23 | "Unable to find motion correction pattern. Pattern match score is below the threshold." |
| 99 | No Error |

public bool **AuthorizationFlag**

Authorization state code

| **Value** | **Meaning** |
|---|---|
| True | Authorization finished |
| False | Authorization not done yet |

public bool **SetUpFlag**

Specifies whether the template image is in place

| **Value** | **Meaning** |
|---|---|
| True | All input images are in place |
| False | Some input images are missing or have error |

**Function:**

public bool **AuthorizationCheck**(string ipAddress)

Connect to LSS controller and unlock the functions.

**Parameter:**

| | |
|---|---|
| ipAddress | IP Address of LSS controller<br>* remove the 0 in front of number. For example, 192.168.0.2, not 192.168.000.002. |

**Return Value:**

| | |
|---|---|
| True | Success |
| False | Fail |

**public void ResetFlag()**

Reset the SetUpFlag (flag whether all input images are in place) to false. Usually called before InputImage functions below.

public bool **InputImageTemp**(int imgWidth, int imgHeight, IntPtr imgBuffer, int roiX, int roiY, int roiWidth, int roiHeight, int noiseRemoval, int featureContrast, bool flagShowOutput, ref IntPtr outputTempImage)

Fill the image buffer of template image and prepare template for matching. When flagShowOutput is set to true, it will output color image (24-bit) with edge aligned to the input image. The size of this image will be the full size of the input image.

**Parameters:**

| | |
|---|---|
| imgWidth | width of image |
| imgHeight | height of image |
| imgBuffer | pointer to image data |
| roiX | x coordinate of ROI |
| roiY | y coordinate of ROI |
| roiWidth | width of ROI |
| roiHeight | height of ROI |
| noiseRemoval | (-1 to 255) set to -1 for automatic setting. Used to remove noise. |
| featureContrast | (-1 to 255) Set to -1 for automatic setting.<br>Note that featureContrast needs to be bigger than noiseRemoval. |
| flagShowOutput | Flag to display template edge image or not<br>When configuring parameters, it is better to see the template edge image (set flagShowOutput to true). However, at run time if you don't need to see the template edge image, configure flagShowOutput to false to save time. |
| outputTempImage | Image pointer of the created template image |

```
public bool InputImageTemp(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer, int roiX, int roiY, int roiWidth, int roiHeight, int noiseRemoval, int featureContrast, bool flagShowOutput, int outImgStep, ref IntPtr outputTempImage)
```

Fill the image buffer of template image and prepare template for matching. When flagShowOutput is set to true, it will output color image (24-bit) with edge aligned to the input image. The size of this image will the full size of the input image.

**Parameters:**

| | |
|---|---|
| imgWidth | width of image |
| imgHeight | height of image |
| imgBuffer | pointer to image data |
| roiX | x coordinate of ROI |
| roiY | y coordinate of ROI |
| roiWidth | width of ROI |
| roiHeight | height of ROI |
| noiseRemoval | (-1 to 255) set to -1 for automatic setting. Used to remove noise. |
| featureContrast | (-1 to 255) Set to -1 for automatic setting.<br>Note that featureContrast needs to be bigger than noiseRemoval. |
| flagShowOutput | Flag to display template edge image or not<br>When configuring parameters, it is better to see the template edge image (set flagShowOutput to true). However, at run time if you don't need to see the template edge image, configure flagShowOutput to false to save time. |
| outputTempImage | Image pointer of the created template image |
| outImgStep | stride of output created template image |
| outputTempImage | Image pointer of the created template image |

```
public bool InputImageTemp(int imgWidth, int imgHeight, IntPtr imgBuffer, int roiX, int roiY, int roiWidth, int roiHeight, int noiseRemoval, int featureContrast)
```

Fill the image buffer of template image and prepare template for matching.

**Parameters:**

| | |
|---|---|
| imgWidth | width of image |
| imgHeight | height of image |
| imgBuffer | pointer to image data |
| roiX | x coordinate of ROI |

| | |
|---|---|
| roiY | y coordinate of ROI |
| roiWidth | width of ROI |
| roiHeight | height of ROI |
| noiseRemoval | (-1 to 255) set to -1 for automatic setting. Used to remove noise. |
| featureContrast | (-1 to 255) Set to -1 for automatic setting<br>Note that featureContrast needs to be bigger than noiseRemoval. |

```
public bool InputImageTemp(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer, int roiX, int
roiY, int roiWidth, int roiHeight, int noiseRemoval, int featureContrast)
```

Fill the image buffer of template image and prepare template for matching.

**Parameters:**

| | |
|---|---|
| imgWidth | width of image |
| imgHeight | height of image |
| imgStep | stride of the input image |
| imgBuffer | pointer to image data |
| roiX | x coordinate of ROI |
| roiY | y coordinate of ROI |
| roiWidth | width of ROI |
| roiHeight | height of ROI |
| noiseRemoval | (-1 to 255) set to -1 for automatic setting. Used to remove noise. |
| featureContrast | (-1 to 255) Set to -1 for automatic setting<br>Note that featureContrast needs to be bigger than noiseRemoval. |

```
public bool InputImageTempEdge(int imgWidth, int imgHeight, IntPtr imgBuffer, int roiX, int roiY, int
roiWidth, int roiHeight, int noiseRemoval, int featureContrast, bool flagShowOutput, ref IntPtr
outputTempImage)
```

Fill the image buffer of template image and prepare template for matching. When flagShowOutput is set to true, it will output black and white (8-bit) image to show the created template. The size of this image will be the same size as ROI.

**Parameters:**

| | |
|---|---|
| imgWidth | width of image |
| imgHeight | height of image |
| imgBuffer | pointer to image data |
| roiX | x coordinate of ROI |
| roiY | y coordinate of ROI |
| roiWidth | width of ROI |
| roiHeight | height of ROI |
| noiseRemoval | (-1 to 255) set to -1 for automatic setting. Used to remove noise. |
| featureContrast | (-1 to 255) Set to -1 for automatic setting<br>Note that featureContrast needs to be bigger than noiseRemoval. |
| flagShowOutput | Flag to display template edge image or not.<br>When configuring parameters, it is better to see the template edge image (set flagShowOutput to true). However, at run time if you don't need to see the template edge image, configure flagShowOutput to false to save time. |
| outputTempImage | Image pointer of the created template image |

```
public bool InputImageTempEdge(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer, int roiX,
int roiY, int roiWidth, int roiHeight, int noiseRemoval, int featureContrast, bool flagShowOutput,
int outImgStep, ref IntPtr outputTempImage)
```

Fill the image buffer of template image and prepare template for matching. When flagShowOutput is set to true, it will output black and white (8-bit) image in specified stride format to show the created template. The size of this image will be the same size as ROI.

**Parameters:**

| | |
|---|---|
| imgWidth | width of image |

| | |
|---|---|
| imgHeight | height of image |
| imgStep | stride of the input image |
| imgBuffer | pointer to image data |
| roiX | x coordinate of ROI |
| roiY | y coordinate of ROI |
| roiWidth | width of ROI |
| roiHeight | height of ROI |
| noiseRemoval | (-1 to 255) set to -1 for automatic setting. Use to remove noise. |
| featureContrast | (-1 to 255) Set to -1 for automatic setting<br>Note that featureContrast needs to be bigger than noiseRemoval. |
| flagShowOutput | Flag to display template edge image or not.<br>When configuring parameters, it is better to see the template edge image (set flagShowOutput to true). However, at run time if you don't need to see the template edge image, configure flagShowOutput to false to save time. |
| outImgStep | stride of output created template image |
| outputTempImage | Image pointer of the created template image |

```
public bool MatchImage(int imgWidth, int imgHeight, IntPtr imgBuffer, int roiX, int roiY, int
roiWidth, int roiHeight, int threshold, bool flagShowOutput, ref IntPtr outputMatchImage, ref int
matchScore, ref int moveDistanceX, ref int moveDistanceY)
```

Fill the image buffer of search image, match the image and output the moveDistanceX and moveDistanceY. If flagShowOutput is true, it will output color image (24-bit) with result aligned to the search image. In search edge image: Green edges represent matching template edge pixels; Red edges represent missing template edge pixels.

**Parameters:**

| | |
|---|---|
| imgWidth | width of image |
| imgHeight | height of image |
| imgBuffer | pointer to image data |
| roiX | x coordinate of ROI |
| roiY | y coordinate of ROI |
| roiWidth | width of ROI |
| roiHeight | height of ROI |
| threshold | threshold to accept the found pattern |
| flagShowOutput | Flag to display search image or not<br>When configuring parameters, it is better to see the template edge image (set flagShowOutput to true). However, at run time if you don't need to see the template edge image, configure flagShowOutput to false to save time. |
| outputMatchImage | Image pointer of the search result image |
| matchScore | the match score (quality of match) |
| moveDistanceX | the x coordinate of movement |
| moveDistanceY | the y coordinate of movement |

```
public bool MatchImage(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer, int roiX, int roiY,
int roiWidth, int roiHeight, int threshold, bool flagShowOutput, int outImgStep, ref IntPtr
outputMatchImage, ref int matchScore, ref int moveDistanceX, ref int moveDistanceY)
```

Fill the image buffer of search image, match the image and output the moveDistanceX and moveDistanceY. If flagShowOutput is true, it will output color image (24-bit) with result aligned to the search image. In search edge image: Green edges represent matching template edge pixels; Red edges represent missing template edge pixels in specified stride format.

**Parameters:**

| | |
|---|---|
| imgWidth | width of image |
| imgHeight | height of image |
| imgStep | stride of image |

| | |
|---|---|
| imgBuffer | pointer to image data |
| roiX | x coordinate of ROI |
| roiY | y coordinate of ROI |
| roiWidth | width of ROI |
| roiHeight | height of ROI |
| threshold | threshold to accept the found pattern |
| flagShowOutput | Flag to display search image or not<br>When configuring parameters, it is better to see the template edge image (set flagShowOutput to true). However, at run time if you don't need to see the template edge image, configure flagShowOutput to false to save time. |
| outImgStep | stride of the output image |
| outputMatchImage | Image pointer of the search result image |
| matchScore | output the match score (quality of match) |
| moveDistanceX | the x coordinate of movement |
| moveDistanceY | the y coordinate of movement |

```
public bool MatchImage(int imgWidth, int imgHeight, IntPtr imgBuffer, int roiX, int roiY, int
roiWidth, int roiHeight, int threshold, ref int matchScore, ref int moveDistanceX, ref int
moveDistanceY)
```

Fill the image buffer of search image, match the image and output the moveDistanceX and moveDistanceY.

**Parameters:**

| | |
|---|---|
| imgWidth | width of image |
| imgHeight | height of image |
| imgBuffer | pointer to image data |
| roiX | x coordinate of ROI |
| roiY | y coordinate of ROI |
| roiWidth | width of ROI |
| roiHeight | height of ROI |
| threshold | threshold to accept the found pattern |
| matchScore | output the match score (quality of match) |
| moveDistanceX | the x coordinate of movement |
| moveDistanceY | the y coordinate of movement |

```
public bool MatchImage(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer, int roiX, int roiY,
int roiWidth, int roiHeight, int threshold, ref int matchScore, ref int moveDistanceX, ref int
moveDistanceY)
```

Fill the image buffer of search image, match the image and output the moveDistanceX and moveDistanceY.

**Parameters:**

| | |
|---|---|
| imgWidth | width of image |
| imgHeight | height of image |
| imgStep | stride of image |
| imgBuffer | pointer to image data |
| roiX | x coordinate of ROI |
| roiY | y coordinate of ROI |
| roiWidth | width of ROI |
| roiHeight | height of ROI |
| threshold | threshold to accept the found pattern |
| matchScore | output the match score (quality of match) |
| moveDistanceX | the x coordinate of movement |
| moveDistanceY | the y coordinate of movement |

```
public bool MatchImageEdge(int imgWidth, int imgHeight, IntPtr imgBuffer, int roiX, int roiY, int
roiWidth, int roiHeight, int threshold, bool flagShowOutput, ref IntPtr outputMatchImage, ref int
matchScore, ref int moveDistanceX, ref int moveDistanceY)
```

Fill the image buffer of search image, match the image and output the moveDistanceX and moveDistanceY. If flagShowOutput is true, it will output black and white (8-bit) image to show the match result. The size of this image will be the same size as ROI.

**Parameters:**

| | |
|---|---|
| imgWidth | width of image |
| imgHeight | height of image |
| imgBuffer | pointer to image data |
| roiX | x coordinate of ROI |
| roiY | y coordinate of ROI |
| roiWidth | width of ROI |
| roiHeight | height of ROI |
| threshold | threshold to accept the found pattern |
| flagShowOutput | Flag to display search image or not<br>When configuring parameters, it is better to see the template edge image (set flagShowOutput to true). However, at run time if you don't need to see the template edge image, configure flagShowOutput to false to save time. |
| outputMatchImage | Image pointer of the search result image |
| matchScore | output the match score (quality of match) |
| moveDistanceX | the x coordinate of movement |
| moveDistanceY | the y coordinate of movement |

```
public bool MatchImageEdge(int imgWidth, int imgHeight, int imgStep, IntPtr imgBuffer, int roiX, int
roiY, int roiWidth, int roiHeight, int threshold, bool flagShowOutput, int outImgStep, ref IntPtr
outputMatchImage, ref int matchScore, ref int moveDistanceX, ref int moveDistanceY)
```

Fill the image buffer of search image, match the image and output the moveDistanceX and moveDistanceY. If flagShowOutput is true, it will output black and white (8-bit) image to show the match result. The size of this image will be the same size as ROI.

**Parameters:**

| | |
|---|---|
| imgWidth | width of image |
| imgHeight | height of image |
| imgBuffer | pointer to image data |
| roiX | x coordinate of ROI |
| roiY | y coordinate of ROI |
| roiWidth | width of ROI |
| roiHeight | height of ROI |
| threshold | threshold to accept the found pattern |
| flagShowOutput | Flag to display search image or not<br>When configuring parameters, it is better to see the template edge image (set flagShowOutput to true). However, at run time if you don't need to see the template edge image, configure flagShowOutput to false to save time. |
| outImgStep | stride of the output image |
| outputMatchImage | Image pointer of the search result image |
| matchScore | output the match score (quality of match) |
| moveDistanceX | the x coordinate of movement |
| moveDistanceY | the y coordinate of movement |

**Use Case**

| | | |
|---|---|---|
|  | Input Images | North Image – light coming from the North direction (top of the part)<br>This is also considered the Template Image for motion correction |
|  | | West Image – Light coming from the West direction (left of the part) |
|  | | South Image – Light coming from the bottom of the part |
|  | | East Image – Light coming from the East direction (right of the part) |
|  | Search Image | Search Image – used to find the previously defined pattern to determine how the part moved in the field of view |

**Application Example in C#**

```
using CCS_CI_SDK;

CI_MCOR myMCOR;
CI_SFS mySFS;

int imgWidth = 2496;
int imgHeight = 2048;
int imgStep = 2496;

Rectangle roiTemplate = new Rectangle(0, imgHeight / 3, imgWidth / 3, imgHeight / 3);
Rectangle roiSearch = new Rectangle(0, imgHeight / 3 - 20, imgWidth, imgHeight / 3 + 40);
Rectangle roiMerge = new Rectangle(imgWidth / 6, imgHeight / 5, imgWidth * 2 / 5,
imgHeight * 3 / 5);

int scoreThreshold = 30;
int myScore = 0;
int myMoveX = 0;
int myMoveY = 0;
int myNoiseRemoval = -1;
int myFeatureSize = -1;
```

```
bool patternSuccess = false;
bool matchSuccess = false;
bool shapeSuccess = false;
bool textureSuccess = false;

int myKernal = 1;
int myBrightness = 3;
int myContrast = 0;

IntPtr ptrInputImageNorth = ptrOriginal1;  // pointer to 2496x2048 image data
IntPtr ptrInputImageWest = ptrOriginal2;
IntPtr ptrInputImageSouth = ptrOriginal3;
IntPtr ptrInputImageEast = ptrOriginal4;
IntPtr ptrInputImageSearch = ptrOriginal5;
IntPtr ptrMergeImageShape = new IntPtr();
IntPtr ptrMergeImageTexture = new IntPtr();

myMCOR = new CI_MCOR();
mySFS = new CI_SFS();

//Authorization the default IP address of LSS is 192.168.0.10
mySFS.AuthorizationCheck("192.168.0.10");
myMCOR.AuthorizationCheck("192.168.0.10");

//Run MCOR class to find out the movement between each image.
myMCOR.ResetFlag();
patternSuccess = myMCOR.InputImageTemp(imgWidth, imgHeight, imgStep, ptrInputImageNorth,
roiTemplate.X, roiTemplate.Y, roiTemplate.Width, roiTemplate.Height, myNoiseRemoval,
myFeatureSize);
if (patternSuccess)
       matchSuccess = myMCOR.MatchImage(imgWidth, imgHeight, imgStep, ptrInputImageSearch,
       roiSearch.X, roiSearch.Y, roiSearch.Width, roiSearch.Height, scoreThreshold, ref
       myScore, ref myMoveX, ref myMoveY);
//Check the error code of myMCOR.MCORError here

if (!matchSuccess)
       return;

//If MCOR class success, run SFS class to create the final shape image and texture image.
mySFS.ResetFlag();
mySFS.InputImageNorth(imgWidth, imgHeight, imgStep, ptrInputImageNorth);
mySFS.InputImageWest(imgWidth, imgHeight, imgStep, ptrInputImageWest);
mySFS.InputImageSouth(imgWidth, imgHeight, imgStep, ptrInputImageSouth);
mySFS.InputImageEast(imgWidth, imgHeight, imgStep, ptrInputImageEast);

if (mySFS.SetUpFlag)
{
       shapeSuccess = mySFS.MergeShapeSFS(myKernal, myBrightness, myContrast, roiMerge.X,
       roiMerge.Y, roiMerge.Width, roiMerge.Height, myMoveX/4, myMoveY/4, ref
       ptrMergeImageShape);
       textureSuccess = mySFS.MergeTextureSFS(roiMerge.X, roiMerge.Y, roiMerge.Width,
       roiMerge.Height, myMoveX/4, myMoveY/4, ref ptrMergeImageTexture);
}
//Check the error code of mySFS.SFSError here
```

## Required Components

To work with the Shape from Shaping SDK, you will need the following items:

- LSS-2404 Controller
- LSS-2404 Web-based software
  - The LSS software can be accessed via the link below
    - http://bit.ly/LSSCI
  - Note: the LSS software works with a Firefox browser (version 56 or above) or Microsoft Edge
- Compatible Lights (e.g. segmented ring light or bar light sets) and corresponding cables
- Compatible Camera
- Machine vision development software (optional)

## Technical Support

Limited technical support for the SDK is available by contacting CCS at techsupport@ccsamerica.com

**Issue 1.0 – February 2019**



CCS America, Inc.

6 Lincoln Knoll Lane

Suite 102

Burlington, MA 01803

T: 781-272-6900

F: 781-272-6902

CCS Japan Inc.

374 Okakuencho, Shimodachiuri-agura,

Karasuma-dori, Kamigyo-ku

Kyoto, 602-8011, Japan

T: +81-75-415-8277

F: +81-75-415-8278