755 Phoenix Drive, Ann Arbor MI, 48108, USA
Phone: 734 668-2930 • Fax: 734 668-2877 • Email: info@carsim.com
carsim.com

VS Camera Sensor Simulink Block

This memo describes the Camera Sensor Simulink block, available for use with the VehicleSim (VS) products BikeSim, CarSim, and TruckSim. This memo will discuss usage with CarSim.

The Camera Sensor block is an S-function block for use within Simulink. This block is available in the S-functions provided with BikeSim, CarSim, and TruckSim. Figure 1 shows the block provided with CarSim.

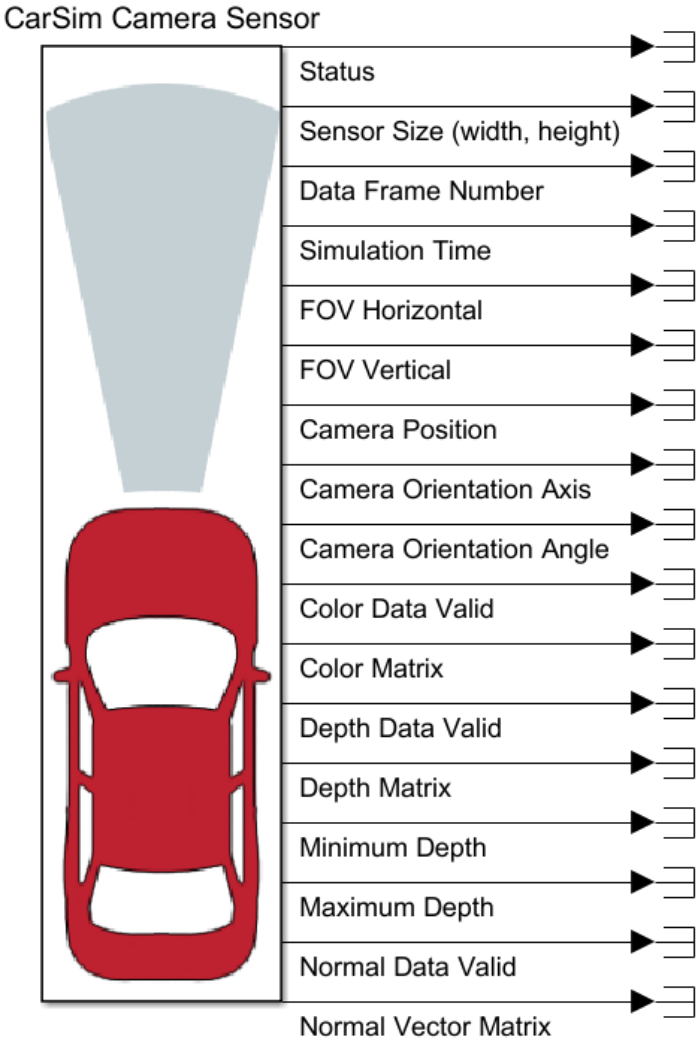

Figure 1. Simulink S-function block using the VsSensor MEX file.

The Camera Sensor block uses the VsSensor S-function to supply your Simulink model with Camera Sensor data generated by VS Visualizer. The VsSensor S-function is located in the MEX

files `VsSensor.mexw32` and `VsSensor.mexw64`, which can be found within the CarSim installation directory (`CarSim_Prog\Programs\solvers\Matlab84+`).

Operation Overview

Shared system memory is a means by which two programs can access the same area of computer memory. The VS Shared Buffer system used by VS Visualizer and VsSensor S-function utilizes shared system memory to transfer rendered image data from VS Visualizer to the VsSensor S-function for use within Simulink.

The VsSensor S-function uses the VS Shared Buffers interface to retrieve rendered image data from VS Visualizer and make that data available for processing within Simulink. This interface provides a mechanism for separate processes (programs) to share data with minimal overhead, while providing robust inter-process synchronization.

The VsSensor S-function can provide three types of render data for every pixel of each rendered frame: RGB color, depth, and surface normal vector.

S-function Block Parameters

To use the Camera Sensor block, specify the name of a Shared Buffer in the `Shared Buffer Name` field of the Block Parameters dialog (Figure 2).

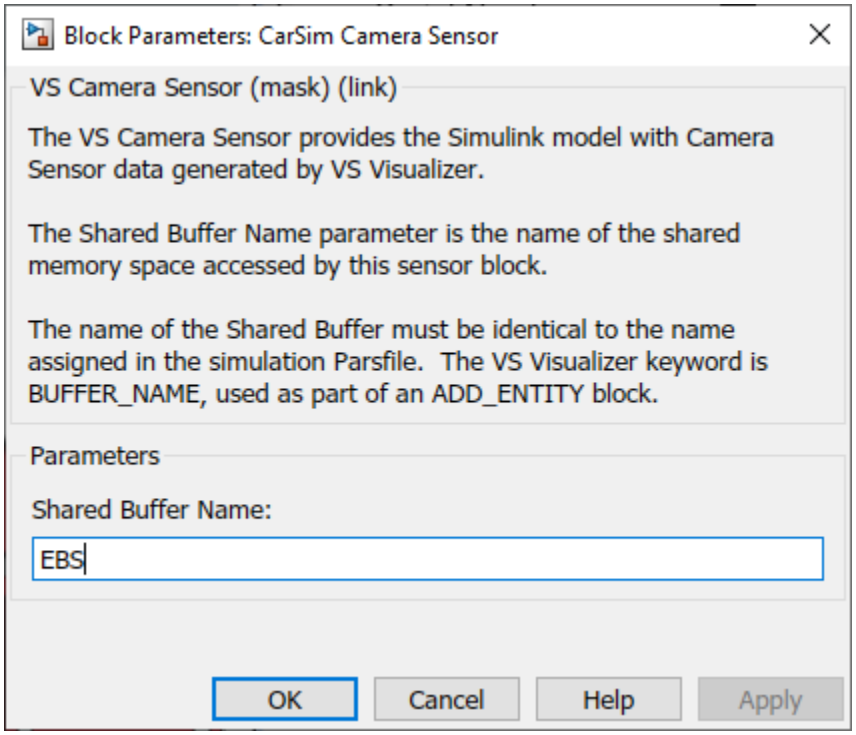

Figure 2. CarSim Camera Sensor Block Parameters.

Note that the Shared Buffer name must be identical to the name given to the Shared Buffer in the Parsfile loaded by VS Visualizer (Listing 1).

Listing 1. Example VS Visualizer Parsfile.

```
! Create a new Cameraman (camera) named "My Cameraman", attached to a
! Reference Frame or Transform named "Camera Location":
ADD_ENTITY Cameraman Fixed
  NAME My Cameraman
  BASE_REFERENCE_FRAME Camera Location
  SET_FIELD_OF_VIEW 55
END_ENTITY Cameraman Fixed

! Create a 400x225 pixel Render View named "My Render View" and
! attach it to the Cameraman "My Cameraman":
ADD_ENTITY RenderView My Render View
  CAMERAMAN_NAME My Cameraman
  TEX_W 400
  TEX_H 225
END_ENTITY RenderView My Render View

! Create a new Shared Buffer named "My Sensor" and attach it
! to the Render View named "My Render View", and share three types
! of data: RGB color, depth, and normal vectors:
ADD_ENTITY SharedBuffer
  BUFFER_NAME My Sensor
  BUFFER_RENDERVIEW My Render View
  BUFFER_ADD_CONTENT_TYPE rgb
  BUFFER_ADD_CONTENT_TYPE depth
  BUFFER_ADD_CONTENT_TYPE normal
END_ENTITY SharedBuffer
```

If the specified Shared Buffer is not available, the S-function will report an error (Figure 3).

This error may occur for several reasons, including:

- VS Visualizer is not running.
- VS Visualizer is not configured to create the Shared Buffer.
- The Shared Buffer name as specified as the S-function parameter differs from that which was specified with the `BUFFER_NAME` keyword in the VS Visualizer Parsfile. The Shared Buffer name is case-sensitive and must be **exactly** the same in both places (including punctuation and spaces).

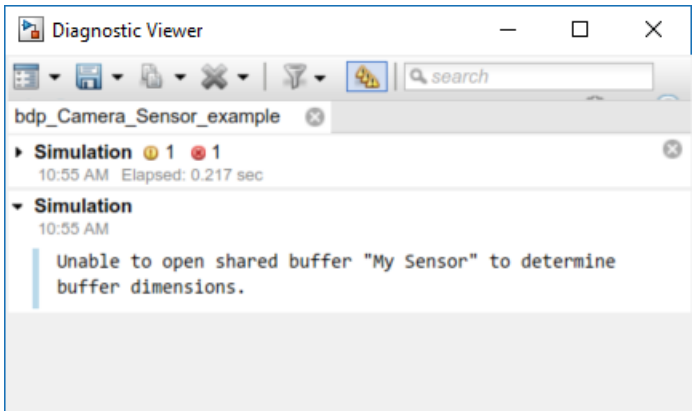

Figure 3. Error message from VsSensor when it cannot locate the specified Shared Buffer.

Inter-process Synchronization and Data Atomicity

The VS Shared Buffer system operates with a client-server model; VS Visualizer is the "server" that creates the Shared Buffer, and the VsSensor S-function is the "client" that connects to an existing Shared Buffer (that was created by VS Visualizer). The server, VS Visualizer, is considered the owner of the Shared Buffer.

While two-way communication is possible in the client-server model, the VS Shared Buffer system only facilitates one-way communication as of the VehicleSim 2017 release. Only the server component (VS Visualizer) may write data to the Shared Buffer, while the client is granted read-only access to the data.

Care must be taken to ensure that the two programs respect one another when reading or modifying data in this shared area. VS Visualizer and the VsSensor S-function cooperate to handle this automatically. This inter-process synchronization works because the server and the client are aware of when the other is accessing the data. Each will wait until the other is finished before accessing the data themselves. For example, if VS Visualizer is ready to write a new frame of data to the Shared Buffer, it must wait until VsSensor has completed copying data from the previous frame. Otherwise, it would be possible for VsSensor to read part of the frame data (the Color buffer, for example) from one frame, and other data (such as the depth buffer) from a subsequent frame. The VS Shared Buffer system ensures that this cannot happen. For a given time step in Simulink, it is guaranteed that all outputs of the VsSensor S-function belong to the same frame of data provided by VS Visualizer.

Inter-process synchronization does come at a cost. VS Visualizer cannot begin to render the next frame while VsSensor is reading the shared memory area, and VsSensor cannot start reading the shared memory area while VS Visualizer is in the process of writing new data to the shared memory area. This is known as "blocking" and can decrease the performance of both programs.

Due to the asynchronous communication between the CarSim solver, VS Visualizer, and the VS Sensor S-function, the Simulink model does not control exactly when a frame is rendered. The VsSensor S-function output **Data Frame Number** is provided to help manage this. This output indicates to your Simulink model when a frame of new data is available.

Your Simulink model may run at a much higher rate than VS Visualizer is rendering. For example, your Simulink model may run at 1000 Hz and contain a CarSim Solver block that is

configured to send data to VS Visualizer for Live Animation at 100 Hz. This means that the data coming out of the Camera Sensor block will only change approximately every 10 time steps, and the data that is received will be delayed one or more time steps. The **Data Frame Number** output makes it possible to detect exactly when the new frame data becomes available.

Output Data Relationships

The three sensor output matrices (**Color Matrix**, **Depth Matrix**, and **Normal Vector Matrix**) are directly related. The data in these matrices are from the same rendered frame, and the individual elements in each output correspond to those in the other output matrices. For example, if you are interested in the pixel at location [22,33], the **Color Matrix** output at this location is a three-dimensional vector containing the RGB color of that pixel, the same location in the **Depth Matrix** output contains the depth of the object rendered in that pixel, and the **Normal Vector Matrix** output contains a three-dimensional vector that is the normal vector of the surface rendered at that same pixel. The first two dimensions of these three matrices will always be the same size (the height and width of the rendered frame).

S-function Block Output Reference

Table 1 lists the output vectors provided by the Camera Sensor block. The output vectors are identified in Simulink by index. The names shown in the table are used in this document but can be changed in Simulink if desired.

Table 1. Camera Sensor S-function outputs

Index	Name	Data Type	Size	Description
0	Status	boolean	[1]	Indicates that Shared Buffer is connected.
1	Sensor Size	integer	[2]	Width and height of the Color, Depth, and Normal Vector matrices (pixels).
2	Data Frame Number	integer	[1]	Indicates when incoming data has changed.
3	Simulation Time	double	[1]	Simulation time at which the sensor data was generated.
4	FOV Horizontal	double	[1]	Horizontal Field of View (radians).
5	FOV Vertical	double	[1]	Vertical Field of View (radians).
6	Camera Position	double	[3]	X, Y, Z position of the camera sensor in world coordinates.
7	Camera Orientation Axis	double	[3]	Axis-Angle representation of the orientation of the camera in world coordinates. Angle is in radians.
8	Camera Orientation Angle	double	[1]	
9	Color Data Valid Flag	boolean	[1]	Indicates if the Color Matrix output contains valid data.
10	Color Matrix	integer	[h][w][3]	Matrix containing RGB color data in 8-bit-per-pixel format for each pixel.
11	Depth Data Valid Flag	boolean	[1]	Indicates if the Depth Matrix output contains valid data.
12	Depth Matrix	single	[h][w]	Matrix containing a depth value for each pixel.
13	Minimum Depth	single	[1]	Lowest depth value present in the Depth Matrix.
14	Maximum Depth	single	[1]	Highest depth value present in the Depth Matrix.
15	Normal Vectors Valid Flag	boolean	[1]	Indicates if the Normal Vector Matrix output contains valid data.
16	Normal Vector Matrix	single	[h][w][3]	Matrix containing a 3-dimentional (X,Y,Z) vector for each pixel which identifies the surface normal direction relative to the camera's view direction.

Status

This is a boolean output that specifies if the underlying Shared Buffer is connected to a data source (such as VS Visualizer). If this output is 1, it means that it is connected and active (receiving data). If this output is 0, it means that the block is not connected and there is no new data coming in.

Sensor Size

This is a two-dimensional vector which indicates the width and height (in that order) in pixels of the images that are being provided by VS Visualizer. The size reported by this output is identical

to the size of the first two dimensions of the **Color Matrix**, **Depth Matrix**, and **Normal Vector Matrix** outputs.

Data Frame Number

This number is incremented each time a new frame of data is available. The number itself is not meaningful, it is only useful for detecting the arrival of new data.

Simulation Time

This is the simulation time within VS Visualizer of the frame. This output is only valid when VS Visualizer is rendering an off-line (pre-recorded) Run. When using Live Animation, this number will not necessarily correspond with the simulation time in Simulink. This limitation may be addressed in a future update of this software.

FOV Horizontal

The horizontal field of view of the rendered data, in radians.

FOV Vertical

The vertical field of view of the rendered data, in radians.

Camera Position

This is a vector of three values: X, Y, and Z position of the camera location in world coordinates (using the VehicleSim coordinate system).

Camera Orientation Axis

This value combined with **Camera Orientation Angle** represent the orientation of the camera in axis-angle format. This output contains a vector of three values: X, Y, and Z. These components represent the rotation axis in world coordinates (using the VehicleSim coordinate system).

Camera Orientation Angle

This value combined with **Camera Orientation Axis** represent the orientation of the camera in axis-angle format. This output contains the rotation angle in radians about the axis.

Color Data Valid Flag

This boolean flag indicates if the **Color Matrix** output contains valid data. VS Visualizer can be configured to generate any combination of color, depth, and normal vector outputs. For example, if VS Visualizer is configured to generate the Color and Normal Vector output only, this flag and **Normal Vectors Valid Flag** will both be 1, while **Depth Data Valid Flag** will be 0. This means there may be one type of data available (e.g. Color), while another is not (e.g. Depth).

Color Matrix

This is a three-dimensional matrix of integers containing color information in 8-bit unsigned integers. The first two dimensions are the coordinates of the pixel, the last dimension is the color component (0 = Red, 1 = Green, 2 = Blue). For example, the value at [23, 55, 0] is an integer in

the range of 0 to 255 which specifies how much red the pixel at coordinates [23, 55] contains. [23, 55, 1] contains the amount of green in that same pixel. Not surprisingly, [23, 55, 2] identifies the amount of blue. The pixel coordinates follow the MATLAB/Simulink standard image coordinate system.

Depth Data Valid Flag

This boolean flag indicates if the **Depth Matrix** output contains valid data. See **Color Data Valid Flag** above for more information.

Depth Matrix

This is a two-dimensional matrix containing 32-bit floating point values. Each entry in this matrix indicates the depth from the location of the sensor camera to the surface rendered at that pixel. The depth is in world length units, which is meters for all VehicleSim products.

Minimum Depth

This output specifies the lowest value present in the Depth Matrix.

Maximum Depth

This output specifies the highest value present in the Depth Matrix.

Normal Vectors Valid Flag

This boolean flag indicates if the **Normal Vector Matrix** output contains valid data. See **Color Data Valid Flag** above for more information.

Normal Vector Matrix

This is a three-dimensional matrix of single-precision (32-bit) floating point numbers. For each pixel location, the third dimension contains a vector (X, Y, Z) indicating the surface normal direction relative to the sensor camera's view direction. This vector is in VehicleSim coordinates, with the +X direction corresponding to the "forward" view direction. For example, the vector (-1, 0, 0) is a vector pointing backwards (directly at the camera), and indicates that the surface is perpendicular to the direction in which the camera is aimed.