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

VS Path Detectors

CarSim, TruckSim, and BikeSim support *VS Path Detectors* that generate clothoid (Euler spiral) paths in the immediate vicinity of the lead unit. This is done by looking ahead at a specified VS Reference Path and possibly an LTARG lateral target function that provides a lateral offset relative to the path. Alternatively, the detector can work with three pairs of imported global X-Y coordinates.

Each VS Path Detector fits a cubic polynomial to three target points to create an instantaneous preview path. The detector generates path outputs include curvature, rate of change of curvature, and lateral offset of the path relative to the vehicle. To do this, the VS Path Detector uses a Kalman filter to estimate the curve fit based on noisy target point measurements.

The sole function of a VS Path Detector is to calculate outputs that might be useful for user-defined controllers. The intent is to provide controls engineers with familiar outputs when designing systems such as lane keeping assist systems. In this case, the machine vision system is a “black box” that produces useful outputs (polynomial coefficients, curvature, path offset, etc.).

> **Note** The algorithm used by the VS Path Detector is based on one derived for highway boundary detection, with relatively large radii of curvature and vehicle speeds (approximately 70 km/h or higher). It is not suited for low speed maneuvering. While it will produce outputs regardless, those outputs will not necessarily track the path reliably at low speeds.

Path Detector Variables

By default, a VS Math Model does not have any VS Path Detectors installed. Up to 20 VS Path Detectors may be installed with the command `INSTALL_PATH_DETECTOR`, followed by the number of detectors to be installed. Figure 1 show a portion of an Echo file for a simulation in which a single detector was installed. In this case, none of the parameters were adjusted so all values shown are defaults.

LastRun_ECHO.PAR #

```
!---------------------------------------------------------------------------------
! PATH DETECTOR
!---------------------------------------------------------------------------------
INSTALL_PATH_DETECTOR 1 ! VS Command to install 1 path detector(s)

OPT_PATH_PD(1)        1 ! [D] = 1 -> path detector uses a VehicleSim reference path, =
                        ! 0 -> path detector uses imported global X-Y targets
PATH_ID_PD(1)         1 ! [D] PATH_ID defining path to be detected
LTARG_ID_PD(1)        1 ! [D] LTARG_ID defining path to be detected
VEH_ID_PD(1)          1 ! [D] ID of vehicle in which detector is installed [I]
X_DETECTOR_SM(1)      0 ; mm ! [D] Coordinate of path detector origin in the [x]
                        ! direction relative to the vehicle's sprung mass origin
Y_DETECTOR_SM(1)      0 ; mm ! [D] Coordinate of path detector origin in the [y]
                        ! direction relative to the vehicle's sprung mass origin
L_PVW_PD(1)          20 ; m ! [D] Total lookahead (preview) distance along reference
                        ! path
CURVATURE_STDEV(1)    1 ; 1/m ! [D] Estimated standard deviation of detected curvature
CURV_RATE_STDEV(1)    1 ; 1/m2 ! [D] Estimated standard deviation of detected
                        ! curvature rate
Y_OFFSET_STDEV(1)     1 ; m ! [D] Estimated standard deviation of detected lateral
                        ! offset at path detector origin
Y_MEAS_STDEV(1)       1 ; m ! [D] Estimated standard deviation of measured lateral
                        ! offset at each target point

```

Figure 1. Path detector information visible in an Echo file; default values are shown here.

Note The `INSTALL_PATH_DETECTOR` command may only be used once in a simulation run. VS Math Models support up to separate 20 VS Path Detectors, but they must all be installed with one command. The VS Browser screen described in the next section (page 4) supports up to five detectors. If more are required, they must be defined using generic text input such as done with the **Generic VS Commands** library.

Parameters

The following parameters are defined for each VS Path Detector. All are indexed, using the system index parameter `IDETECTOR`.

- `OPT_PATH_PD` establishes whether the detected path is a VS Reference Path (= 1, the default) or the detected path is given by imported global coordinates (= 0).
- `PATH_ID_PD` and `LTARG_ID_PD` define the path to be detected (not used if `OPT_PATH_PD = 0`). The path to be detected is essentially a base path given by the path ID plus a lateral offset given by the lateral ID.
- `X_DETECTOR_SM` and `Y_DETECTOR_SM` establish the origin of the path detector coordinate system in the sprung mass of the vehicle lead unit. The path detector coordinate system uses the intermediate axis system for its axis directions. These are the axes directions reached by rotating the Earth axis system by `Yaw` about its z-axis. For CarSim and TruckSim, `VEH_ID_PD` defines which vehicle's lead unit is used.

- L_PVW_PD establishes how far ahead of the vehicle the path is detected. In particular, the path segment considered by the path detector is that from the path detector's origin station to the path detector's origin station plus L_PVW_PD. (Station is that of the reference path defined by PATH_ID_PD.) Within this path segment, the path detector takes three target point measurements: at the path detector origin, at the max lookahead point, and halfway in between. If OPT_PATH_PD = 0, L_PVW_PD is not used, and the three target points are instead imported.
- The parameter Y_MEAS_STDEV is a Kalman filter tuning parameter which can be interpreted as the standard deviation of the lateral offset of each target point.
- CURVATURE_STDEV, CURV_RATE_STDEV, and Y_OFFSET_STDEV are Kalman filter tuning parameters which can be interpreted as the standard deviations of the detected curvature at the path detector origin, the detected curvature change rate over the path segment, and the detected lateral offset of the path at the path detector origin.

Output Variables

The outputs installed with each path detector instance are shown in Table 1. In the names, the index *i* indicated the detector index (1, 2, …).

Table 1. Outputs available when path detector is installed.

Output Variable	**Units**	**Description**
X_Detector_*i*, Y_Detector_*i*	m	Global (Earth) coordinates of path detector *i*'s origin.
XpdTarg*j*_*i*, YpdTarg*j*_*i*	m	Path detector coordinates of the measured (or imported) target point *j*, $j = 1,2,3$, for path detector instance *i*.
CurvRatePD_*i*	$1/m^2$	Curvature change rate of the fitted curve associated with path detector instance *i* (derivative of curvature with respect to path detector *i*'s x-coordinate).
CurvPD_*i*	1/m	Curvature of the fitted curve associated with path detector instance *i* at path detector *i*'s origin.
YoffsetPD_*i*	m	Lateral offset of the fitted curve associated with path detector instance *i* at path detector *i*'s origin.

The fitted curve associated with path detector instance *i*, in path detector *i* coordinates, is:

$$y = \frac{\texttt{CurvRatePD\_i}}{6}x^3 + \frac{\texttt{CurvPD\_i}}{2}x^2 + \texttt{YoffsetPD\_i} \qquad \textit{Equation 1}$$

This fitted curve is based on the target points but does not necessarily interpolate them exactly. As this is a clothoidal approximation, curvature in the path detector *i* coordinate system is given by:

$$\kappa = (\texttt{CurvRatePD\_i})x + \texttt{CurvPD\_i} \qquad \textit{Equation 2}$$

To go from path detector coordinates (x, y) to global coordinates (X, Y), you may use the following equations:

$$\mathrm{X} = \mathrm{x}\cos \texttt{Yaw} - \mathrm{y}\sin \texttt{Yaw} + \texttt{X\_Detector\_}i$$

$$Y = x \sin \mathtt{Yaw} + y \cos \mathtt{Yaw} + \mathtt{Y\_Detector\_}i\,.$$

where `Yaw` is the yaw angle of the vehicle the detector is installed into.

Import Variables

Users may bypass VS Reference Paths by importing the three target points in global coordinates. In this case, set `OPT_PATH_PD = 0`, and note that the parameters that define the VS path, `PATH_ID_PD`, `LTARG_ID_PD`, and `L_PVW_PD` are unused. Instead, the three target points are given by

- `IMP_X_PDTARG`*j*`_`*i*. Global coordinates of the imported target point j, $j = 1{,}2{,}3$, for path detector instance i.
- `IMP_Y_PDTARG`*j*`_`*i*. Global coordinates of the imported target point j, $j = 1{,}2{,}3$, for path detector instance i.

These imports are intended to be used only in a replace mode (`REPLACE`, `VS_REPLACE`).

Screen: Path Detectors for External Driver Control

The path detector may be installed from the graphical user interface using the library **Path Detectors for External Driver Control**, located in the category **Custom Forces and Motion Sensors**. Figure 2 shows the screen and identifies the user controls.

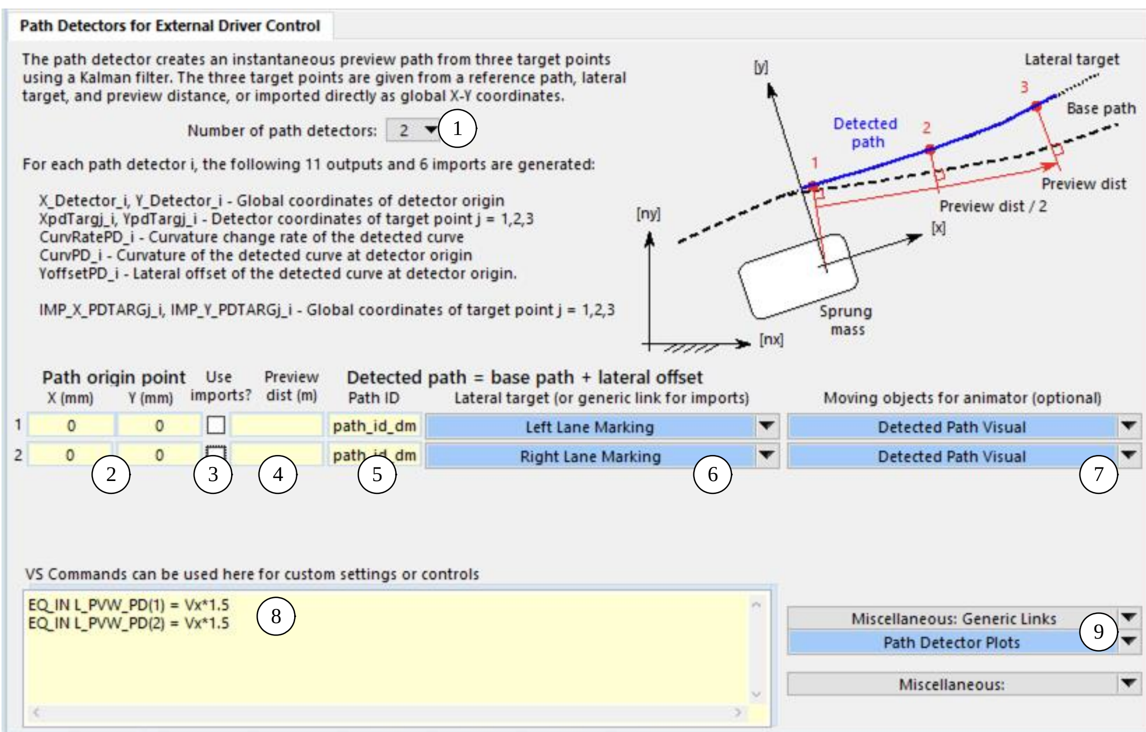

Figure 2. The path detector screen.

① Dropdown to install the selected **number of path detectors**. The math model keyword is `INSTALL_PATH_DETECTORS`. Up to five may be installed using this screen, suitable for the lane markings of a four-lane road.

② **Origin in sprung mass** (indexed keywords `X_DETECTOR_SM` and `Y_DETECTOR_SM`). Locates the origin of the path detector coordinate system in the sprung mass of the lead unit. The path detector coordinate system uses the intermediate axis system for its axis directions (denoted by [x] and [y]). These are the axes directions reached by rotating the Earth axis system by `Yaw` about its z-axis. Earth's axis system is denoted by [nx] and [ny].

③ **Use imports?** When unchecked, indexed math model keyword `OPT_PATH_PD = 1` and the path is to be specified using a path ID and lateral target. When this box is checked, `OPT_PATH_PD = 0`, and the lateral target blue link ⑥ is replaced with a generic VS commands link suitable for setting up the import variables for the three target points. Additionally, when the box is checked, the preview distance ④ and path ID ⑤ are hidden.

④ **Preview distance**, indexed math model keyword `L_PVW_PD`. This establishes how far ahead of the vehicle the path is measured. Referring to the diagram on Figure 2, the path detector takes three target point measurements: at the path detector origin station, at the max lookahead point, and halfway in between. The distances are all in terms of path station. This length is not needed when imports are used, although it still exists and can be repurposed for visualizations.

⑤ **Path ID**, indexed math model keyword `PATH_ID_PD`. Defines the base path to be detected. This path is assumed to be defined elsewhere, perhaps associated with a road. In Figure 2 it is set to `PATH_ID_DM`, which is the path ID used by the driver model.

⑥ **Lateral target blue link** or **generic VS commands blue link**, depending on the checkbox ③. If imports are not used, use to link to a lateral target table defined using a **Control: Steering by the Closed-loop Driver Model** screen. The intent is to easily define lane markings and road boundaries relative to the base path. This works by setting the indexed math model keyword `LTARG_ID_PD` to match that defined by the lateral target table. (On the closed-loop driver model screen, be sure to uncheck "Use closed-loop steering and show parameters".) If imports are used, use ⑥ to instead link to a generic VS commands dataset which sets up the imports.

⑦ **Moving objects for animator**. Optional link to **Multiple Moving Objects** dataset which defines a visualization of the detected path. The example database included with your VehicleSim product includes one such visualization "Detected Path Visual". This uses the symbol stack variable <<detector>>, which is set by the path detectors screen, to create the necessary outputs for animating a series of dots.

⑧ **Miscellaneous yellow field** suitable for custom settings or controls. In this case, the indexed `L_PVW_PD` parameters have been set to vary with vehicle speed by way of a preview time of 1.5 seconds, which is why no value is entered in control ④.

⑨ **Miscellaneous links** which are suitable for plots, more custom settings, etc. Note that this field has *splitter* controls on the top and right edge, allowing the fields to be stretched in case it includes more lines of text or if the lines are too long to view with the original width.

Technical Details

The primary technical reference for the path detector implementation is the technical paper *Model based detection of road boundaries with a laser scanner* (Kirchner & Heinrich, 1998). However, the implementation here is not identical, as we do not have to worry about the implications of physical measurements. Coordinate systems also differ, and the task is different in the sense that we are detecting a VehicleSim reference path rather than road boundaries. The notation and equations used here for the Kalman filter also differ from Kirchner & Heinrich slightly, to better match popular references (Wikipedia, 2020).

> **Note** To simplify notation, we drop the index *i* associated with each path detector instance.

Predicted Model State

The Kalman filter state vector is

$$\boldsymbol{x}(k|k) = (\texttt{CurvRatePD}, \texttt{CurvPD}, \texttt{YoffsetPD})^T$$

where the notation (*m*|*n*) means the value at timestep *m* using information from timestep *n*. At the beginning of the timestep *k*, all that is available is the value from the last timestep, $\boldsymbol{x}(k-1|k-1)$.

The state vector is initialized as the zero vector.

The value of the state vector at timestep *k* using information from previous timestep *k-1* is predicted as

$$\boldsymbol{x}(k|k-1) = \begin{pmatrix} 1 & 0 & 0 \\ \delta_x & 1 & 0 \\ \delta_x^3/6 & \delta_x^2/2 & 1 \end{pmatrix} \boldsymbol{x}(k-1|k-1) + \begin{pmatrix} 0 \\ 0 \\ -\delta_y \end{pmatrix} =: \boldsymbol{F}(k)\boldsymbol{x}(k-1|k-1) + \boldsymbol{g}(k),$$

where (δ_x, δ_y) is the path detector origin's displacement from the previous timestep to the current timestep (in path detector coordinates).

The model state uncertainty is a 3x3 matrix denoted by $\boldsymbol{P}$, which is initialized to the identity matrix. The model uncertainty at timestep *k* using information from timestep *k*-1 is

$$\boldsymbol{P}(k|k-1) = \boldsymbol{F}(k)\boldsymbol{P}(k-1|k-1)\boldsymbol{F}(k)^T + \boldsymbol{Q}(k)$$

where $\boldsymbol{Q}$ denotes the model state covariance matrix, calculated as

$$\boldsymbol{Q}(k) = \begin{pmatrix} \texttt{CURV\_RATE\_STDEV}^2 & 0 & 0 \\ 0 & \texttt{CURVATURE\_STDEV}^2 & 0 \\ 0 & 0 & \texttt{Y\_OFFSET\_STDEV}^2 \end{pmatrix}$$

Measurements

The measurement vector is

$$\boldsymbol{z}(k) = (\texttt{YpdTarg1}, \texttt{YpdTarg2}, \texttt{YpdTarg3})^T$$

The value of the measurement as predicted by the predicted model state is

$$\boldsymbol{z}(k|k-1) = \begin{pmatrix} \texttt{XpdTarg1}^3/6 & \texttt{XpdTarg1}^2/2 & 1 \\ \texttt{XpdTarg2}^3/6 & \texttt{XpdTarg2}^2/2 & 1 \\ \texttt{XpdTarg3}^3/6 & \texttt{XpdTarg3}^2/2 & 1 \end{pmatrix} \boldsymbol{x}(k|k-1) =: \boldsymbol{H}(k)\boldsymbol{x}(k|k-1)$$

The innovation covariance is a 3x3 matrix denoted by $\boldsymbol{S}$; in particular,

$$\boldsymbol{S}(k) = \boldsymbol{H}(k)\boldsymbol{P}(k|k-1)\boldsymbol{H}(k)^T + \boldsymbol{R}(k)$$

where $\boldsymbol{R}$ denotes the measurement covariance matrix

$$\boldsymbol{R}(k) = \texttt{Y\_MEAS\_STDEV}^2 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Updated Model State

The Kalman gain $\boldsymbol{W}$ is then computed as

$$\boldsymbol{W}(k) = \boldsymbol{P}(k|k-1)\boldsymbol{H}(k)^T\boldsymbol{S}(k)^{-1}$$

which is used to produce the final state estimate

$$\boldsymbol{x}(k|k) = \boldsymbol{x}(k|k-1) + \boldsymbol{W}(k)(\boldsymbol{z}(k) - \boldsymbol{z}(k|k-1))$$

and final state uncertainty estimate

$$\boldsymbol{P}(k|k) = (\boldsymbol{I} - \boldsymbol{W}(k)\boldsymbol{H}(k))\boldsymbol{P}(k|k-1)$$

These values are then used as the starting point for the next timestep.

Polynomial Approximation of the Clothoid

The cubic polynomial of *Equation 1* is not mathematically identical to that of a clothoid. The defining characteristic of clothoids is that curvature increases linearly with arc length. If we are to treat the polynomial as such a clothoid, two things need to be true:

1. Path detector coordinate x is equal to arc length.
2. The curvature of the polynomial is equal to *Equation 2*.

Regarding Condition 1, a parametric curve $\alpha(t) \in \mathbb{R}^2$ is parametrized by arc length if $|\alpha'(t)| = 1$ for all t (do Carmo, 1976). In other words, t is the arc length of the curve measured from some starting point on the curve. In this case, the cubic polynomial can be considered a parametric curve of the form:

$$\alpha(t) = \left(t, \frac{\mathrm{a}}{6}t^3 + \frac{\mathrm{b}}{2}t^2 + \mathrm{c}\right) \qquad \textit{Equation 3}$$

with derivative:

$$\alpha'(t) = \left(1, \frac{a}{2}t^2 + bt\right)$$

The length of the derivative vector is then:

$$|\alpha'(t)| = \sqrt{1 + \left(\frac{a}{2}t^2 + bt\right)^2}$$

We see that $|\alpha'(t)| = 1$ for all t if

$$\frac{a}{2}t^2 + bt = 0$$

Now let us consider Condition 2. The curvature of a parametric curve $\alpha(t) = \big(x(t), y(t)\big)$ is (do Carmo, 1976):

$$k(t) = \frac{x'y'' - x''y'}{((x')^2 + (y')^2)^{3/2}}$$

This means the curvature of the curve of *Equation 3* is

$$k(t) = \frac{at + b}{\left(1 + \left((a/2)t^2 + bt\right)^2\right)^{3/2}}$$

Note that this is the same as *Equation 2* if

$$\frac{a}{2}t^2 + bt = 0$$

In other words, Conditions 1 and 2 are satisfied by the same term being equal to zero. In terms of the notation used in the previous sections, the approximation is true when

$$\frac{\texttt{CurvRatePD}}{2}x^2 + \texttt{CurvPD}\,x = 0$$

Thus, the quality of the approximation (of the clothoid as the cubic polynomial) decreases as path detector coordinate x increases; at $x = 0$, the above condition is satisfied.

References

do Carmo, M. P. (1976). *Differential Geometry of Curves and Surfaces.* Englewood Cliffs, New Jersey: Prentice-Hall.

Kirchner, A., & Heinrich, T. (1998). Model based detection of road boundaries with a laser scanner. *1998 IEEE International Conference on Intelligent Vehicles*, (pp. 93-98).

Wikipedia. (2020, January 30). *Kalman filter*. Retrieved from Wikipedia: https://en.wikipedia.org/w/index.php?title=Kalman_filter&oldid=937359378