robótica

Robótica
Departamento de Ingeniería de Sistemas y Automática
Universidad Politécnica de Valencia

EUROPEAN COMMISSION. DIRECTORATE GENERAL - JRC
Joint Research Centre – Ispra
Institute for the Protection and the Security of the Citizen (IPSC)

ipsc

virtualrobot

virtualrobot modeller

virtualrobot simulator

virtualrobot translator

VIRTUAL ROBOT EXTERNAL ACCESS LIBRARY

(SUMMARY)

October 2012, Version 6.7Sb (VREAL 6.7 Summary)

Developed by

robótica
Department of System Engineering and Control (DISA)
Polytechnic University of Valencia (UPV)

Contact Address

vrs@isa.upv.es
http://robotica.isa.upv.es/virtualrobot
DISA - UPV
Camino de Vera s/n E-46022, Valencia (Spain)
Tf: +34-963.879.575 Fax: +34-963.879.579

Supported by

European Commission. Directorate General - JRC
Joint Research Centre - Ispra
Institute for the Protection and the Security of the Citizen (IPSC)

Index

1. Objective

The objective of this document is to explain the main functionalities of a library developed to provide access to some of the functionality of *Virtual Robot Simulator* (*VRS*) from a user application. **The complete functionality is included in VREAL documentation**. The library, called *VR External Access Library* (*VREAL*), is implemented as a Dynamic Link Library on the file "vreal.dll". The user application can manage, through this library, the elements defined on *VRS*.
VREAL can be compiled with a dynamic link on the user application. The "vreal.h" file contains the definition of the library, while the "vreal.lib" includes the library access specification. Both files must be used to compile and link the user application, while the "vreal.dll" file must be accessible for the user application execution.

2. Implementation

The *VR External Access Library* provides an interface formed by a set of functions whose objective is to allow the user to interact with the kernel of *VRS*.
In addition to constant and type definition, there are the following independent sets of functions:

- A set of functions to initialize and close the Library
- A set of functions to manage files
- A set of functions to edit robots and the environment
- A set of functions to define and modify robot parameters
- A set of functions for speed control
- A set of functions for tool handle
- A set of functions for robot motion
- A set of functions for robot attachment
- A set of functions for inputs and outputs
- A set of functions for environment
- A set of functions to handle with robot operation
- A set of functions to handle an auxiliary list of figures
- A set of functions to handle the display options of *VRS*
- A set of functions for video recording

3. Library-Kernel Communication

The communication between the user application and the kernel of *VRS* can be made in local mode or remote mode:

- In local mode, the user application must be run on the same computer than *VRS*. In fact, *VRS* will ask for the name of the user application to be run and will start it. The user application must initialize *VREAL* with the function provided for this (`alInitialize`) with no parameter at all. Then, the user application has access to all the functionality of this library.
- In remote mode, the user application can be run on the same computer than *VRS* or in any other computer. In both cases, a TCP/IP protocol is used for application communication. The user must first activate on *VRS* the VREAL Remote Mode (**File>>Start Remote VREAL Server**). Then the user must start his/her application wherever it is and the user application must initialize *VREAL* with the function provided for this (`alInitialize`) with the appropriate parameter (the name or ip-address of the PC where *VRS* is running). Then, the user application has access to all the functionality of this library.

In any case, **THE LIBRARY MUST BE CLOSED** before the user application finishes.

4. Function Parameters

Some parameters are used in most of the functions and therefore are explained once in this section.
For example, three parameters are used as identifier in many functions:

- `robotId` is the robot identifier. It is returned only by the function `alLoadRobot` and used by any function that applies on a robot.
- `objectId` is the object identifier. It is returned only by the function `alGetObjectId` and used by any function that applies on an object.

- `partId` is the part identifier. It is returned only by the function `alGetPartId` and used by any function that applies on a part.
- `figureId` is the figure identifier. It is returned by any `Add` function and used by any function that applies on a figure.

A location is usually represented in a function be means of:

- The position represented with the values `x,y,z`.
- The orientation represented with the values `alpha, beta` and `gamma`. These three angles are the Euler Angles Type 2 (also called ZYZ).

According to these parameters, a location is computed with the following steps:

1. Rotation of alpha angle related to Z axis
2. Rotation of beta angle related to V axis (Y axis of mobile frame)
3. Rotation of gamma angle related to W axis (Z axis of mobile frame)
4. Translations of x,y,z values related to X, Y, Z axes (axes of fix frame)

There are some options on locations as shown in **VREAL documentation**.

5. Constant Definition

The following constants are defined:

NUM_DOF		NUM_ROBOTS	
NUM_TOOLS	NUM_OBJECTS		NUM_PARTS
RESET		SYNCHRO	
POINTTOPOINT		LINEAR	
ABSOLUTE_MOVEMENT		RELATIVE_MOVEMENT	
ORIGIN	TOOL_FRAME		WORLD
CHECK_RANGE		NO_CHECK_RANGE	
CHECK_ORIENTATION		NO_CHECK_ORIENTATION	
	CLOSEST		
RIGHT_DOWN		RIGHT_UP	
LEFT_UP		LEFT_DOWN	
POSITIVE_WRIST		NEGATIVE_WRIST	
WIRED	SHADED		HIDDEN
ENVIRONMENT_LOADED		ENVIRONMENT_NOT_LOADED	
VISIBLE		INVISIBLE	
ACTIVE_TRACE		NO_ACTIVE_TRACE	
HIDE_TRACE		NO_HIDE_TRACE	
CHECK_COINCIDENCE		NO_CHECK_COINCIDENCE	
TRANSLATION		ROTATION	
X_AXIS	Y_AXIS		Z_AXIS
U_AXIS	V_AXIS		W_AXIS
EULER_ANGLES_TYPE_1	EULER_ANGLES_TYPE_2		EULER_ANGLES_TYPE_3
NUM_DIGITAL_INPUTS		NUM_DIGITAL_OUTPUTS	
NUM_ANALOGICAL_INPUTS		NUM_ANALOGICAL_OUTPUTS	

A color inversion table is shown in **VREAL documentation**.

Their values are defined in one of the following files:

- "Includes\VReal\VRealExternalDefines.h".
- "Includes\VRealCommunications\VRealCommunicationsExternalDefines.h".
- "Includes\VRTransf\VRTransfExternalDefines.h".
- "Includes\VRMaths\VRMathsExternalDefines.h".

All these files are included in "Includes\VReal\VReal.h".

6. Type Definition

In "Includes\GeneralDefines.h", the following types are defined:

```
typedef char STRING[256]
typedef unsigned long COLORREF
```

7. Error Codes

Every function in the library has at least the following possible values to be returned (unless other case stated):

`RET_OK`	Successful execution
`RET_ERROR`	Error in execution

SEE VREAL documentation

8. Initialization Functions

There are two functions to initialize and close the library:

- `alInitialize`
  a) For local mode:
  `int alInitialize()`
  This function initializes the VREAL library in local mode. It must be called before any other function of the library is used.
  b) For remote mode: **SEE VREAL documentation**
- `alClose`
  `int alClose()`
  This function closes the VREAL library. **It must be called before finishing the user application in any of the modes (local or remote).**

9. File Functions

This set of functions mainly allows to load and close robots and environment.

- `alLoadRobot`
  `int alLoadRobot(STRING fileName,int *robotId)`
  This function loads on *VRS* a new robot from a rkf file (indicated on `fileName`) and gives back the robot identifier on `robotID`. When a robot with the same name exists, *VRS* adds a number (1,2,3,…) to the robot name. The maximum number of robots is defined by the constant `NUM_ROBOTS`. The loaded robot becomes the active robot.
- `alCloseRobot`
  `int alCloseRobot(int robotId)`
  This function closes on *VRS* the robot specified with `robotID`. If the closed robot is the active robot, the first robot on the list of robots becomes the active robot.
- `alLoadEnvironment`
  `int alLoadEnvironment(STRING fileName)`
  This function loads an environment (indicated on `fileName`) on *VRS*. As only one environment can be opened on *VRS*, the new environment will replace any other possible environment.
- `alCloseEnvironment`
  `int alCloseEnvironment()`
  This function closes the environment on *VRS*.
- `alCloseAll`
  `int alCloseAll()`
  This function closes all the robots and the environment on *VRS*.

More functions in VREAL documentation

10. Edit Functions

This set of functions mainly allows placing robots and environment.

- `alPlaceRobot`
  ```
  int alPlaceRobot(int robotId,
  double x, double y, double z,
  double alpha, double beta, double gamma)
  ```
  This function places a robot on VRS, that is, specifies where the robot is located in the space. The location is represented with a position (`x,y,z`) and a orientation (`alpha, beta` and `gamma`) specified in Euler Angles type 2.
  Options: **SEE VREAL documentation**
- `alPlaceEnvironment`
  ```
  int alPlaceEnvironment(double x, double y, double z,
  double alpha, double beta, double gamma)
  ```
  This function places the environment on VRS, that is, specifies where the environment is located in the space. The location is represented with a position (`x,y,z`) and a orientation (`alpha, beta` and `gamma`) specified in Euler Angles type 2.
  Options: **SEE VREAL documentation**

More functions in VREAL documentation

11. Robot Definition Functions

This set of functions is mainly designed to obtain and modify the robot configuration.

- `alSetActiveRobot`
  `int alSetActiveRobot(int robotId)`
  This function actives an specific robot specified with the parameter. The information of the active robot is shown on the dynamic information field.
- `alGetActiveRobot`
  `int alGetActiveRobot(int *robotId)`
  This function obtains in its parameter the robot identifier of the active robot.
- `alInvertRobotColor`
  `int alInvertRobotColor(int robotId,int invert)`
  This function inverts the color of the robot specified in the first parameter. The second parameter can be any of the values indicated in the table of color inversions.
- `alGetAvailableRobots`
  ```
  int alGetAvailableRobots(int robotIds[NUM_ROBOTS],
        STRING robotNames[NUM_ROBOTS],
        int *numberOfRobots)
  ```
  This function obtains the arrays of identifiers and associated names of the available robots on *VRS*, that is, the robots loaded. The number of available robots is returned on the last parameter.
  Parameters:
  `robotIds` is an array with the identifiers of the available robots
  `robotNames` is an array of names of the available robots
  `numberOfRobots` is the number of available robots. If it is equal to 0, there is no robot available on *VRS*. It specifies the number of valid values on the arrays.

More functions in VREAL documentation

12. Speed Functions

This set of functions is mainly designed to control speed scale and robot speed.

- `alSetRobotSpeed`
  `int alSetRobotSpeed(int robotId, double robotSpeed)`
  This function sets the speed of the robot given by `robotId` parameter. The `robotSpeed` value must be between 0.0 and 1.0, where 0.0 indicates the slowest speed and 1.0 the fastest speed. The initial default value is 0.5.

More functions in VREAL documentation

13. Tool Functions

This set of functions is mainly designed to manage the tool.

`alSetActiveToolFrame`
`int alSetActiveToolFrame(int robotId,int toolFrameId)`
This function sets the active ToolFrame. The first parameter indicates the robot whose ToolFrame must be changed. The second value indicates the number of the ToolFrame that must be active after the function execution. If the ToolFrame is not defined, the function returns and the active ToolFrame is not changed.

- `alSetActiveTool`
  `int alSetActiveTool(int robotId, int toolId)`
  This function sets the active Tool. The first parameter indicates the robot whose Tool must be changed. The second value indicates the number of the Tool that must be active after the function execution. If the robot does not have this tool, an error is returned.
- `alSetToolStatus`
  `int alSetToolStatus(int robotId,double toolStatus)`
  This function sets the tool status for the active tool. The first parameter indicates the robot whose tool must be changed. The second value indicates the tool status of the active tool of this robot. The value must be in the interval [0.0,1.0], assigning the closest end value in other case.

More functions in VREAL documentation

14. Robot Motion Functions

This set of functions is designed to move robots.

- `alRobotReset`

  ```
  int alRobotReset(int robotId, int resetType)
  ```

  This function moves the robot which identifier is equal to the first parameter to its reset configuration or to its synchronism configuration depending on the value of the second parameter. The value of this second parameter must be `RESET` or `SYNCHRO`.

- `alMoveRobotJoints`

  ```
  int alMoveRobotJoints(int robotId,
        double joints[NUM_DOF])
  ```

  This function moves the robot, which identifier is passed as the first parameter of the function, to the configuration where the values of its joints are the specified in the array that is passed as the second parameter of the function. A linear interpolation on Joint Space is generated for the movement.

- `alGetRobotLocation`

  ```
  int alGetRobotLocation(int robotId,
  double *x, double *y, double *z,
        double *alpha, double *beta, double *gamma)
  ```

  This function obtains the location of the active ToolFrame of the robot given by the `robotId` parameter. ToolFrame location is related to robot Programming Origin frame.
  The location is represented with a position (`x,y,z`) and a orientation (`alpha, beta` and `gamma`) specified in Euler Angles type 2.
  Options: **SEE VREAL documentation**

- `alMoveRobot`

  ```
  int alMoveRobot(int robotId,
  double x, double y, double z,
  double alpha, double beta, double gamma,
  int linearMovement,     int absoluteMovement,
        int frame)
  ```

  This function moves a robot to a given position.
  Parameters:
  The parameter `robotId` is the robot to be moved.
  The `linearMovement` parameter can have two possible values:

`POINTTOPOINT`	For PointToPoint Movement
`LINEAR`	For Linear Movement

  When the value of this parameter is `POINTTOPOINT`, the robot moves from its current location to the destination location without following any special trajectory. When the value of this parameter is `LINEAR`, the robot moves from its current location to the destination location according to a linear trajectory between the two locations for the ToolFrame.
  The `absoluteMovement` parameter can have one of these two values:

`ABSOLUTE_MOVEMENT`	For Absolute movement
`RELATIVE_MOVEMENT`	For Relative movement

  If the value is `ABSOLUTE_MOVEMENT`, it means that the `x, y, z, alpha, beta, gamma` values represent an absolute movement. Otherwise if the value is `RELATIVE_MOVEMENT` the `x, y, z, alpha, beta, gamma` values represent a relative movement from the current location.
  The `frame` parameter can have one of these values:

`ORIGIN`	For Origin Frame
`TOOL_FRAME`	For Tool Frame
`WORLD`	For World Frame

  If the value is `ORIGIN`, the Programming Origin Frame is taken as the reference of the location. If the value of the frame parameter is `TOOL_FRAME`, the current location of the active ToolFrame is taken as the reference frame of the movement. If the value of the frame parameter is `WORLD`, the World Frame is taken as the reference frame of the movement. Therefore, the specified location represents the location of the active

ToolFrame related to the robot programming Origin Frame (for `ORIGIN`), to the current location of the active ToolFrame (for `TOOL_FRAME`) or the world frame (for `WORLD`).
The location is represented with a position (`x,y,z`) and a orientation (`alpha, beta` and `gamma`) specified in Euler Angles type 2.
Options: **SEE VREAL documentation**

- `alMove`

```
int alMove(int robotId,
double x, double y, double z,
double alpha, double beta, double gamma,
      int moveParameter)
```

This function is the same as the previous one but with the motion parameters in just one parameter as addition of them, as in the example:
`alMove(robotId,x,y,z,a,b,g,LINEAR+RELATIVE_MOVEMENT+WORLD)`
`POINTTOPOINT`, `ABSOLUTE_MOVEMENT` and `ORIGIN` are default values and must not be added. None of the parameters can be added twice. The options are the same than in previous function.

- `alApproxToLocation`

```
int alApproxToLocation(int robotId,
double x, double y, double z,
double alpha, double beta, double gamma,
int linearMovement,     int frame,
      double xDistance,double yDistance,
      double zDistance)
```

This function approximates the robot indicated in `robotId` parameter to a location. The actual ToolFrame of the robot will result with the same orientation with the referred location but the position will be the same with an offset of `xDistance, yDistance, zDistance` values related to the robot origin frame (when `frame` is `ORIGIN`) or active ToolFrame (when `frame` is `TOOL_FRAME`) according to the `frame` parameter (`WORLD` cannot be used). The `linearMovement` parameter has the same meaning that in `alMoveRobot`.
The location is represented with a position (`x,y,z`) and a orientation (`alpha, beta` and `gamma`) specified in Euler Angles type 2 **always referred to the robot origin frame**.
Options: **SEE VREAL documentation**

- `alMoveToPart`

```
int alMoveToPart(int robotId, int partId,
int opFrameId,int linearMovement)
```

This function moves the robot indicated in `robotId` parameter to the location of the `opFrameId` of the `partId` parameter, making coincident the actual ToolFrame of the robot with the referred operation frame of the part, both in position and orientation. The `linearMovement` parameter has the same meaning that in `alMoveRobot`.

- `alApproxToPart`

```
int alApproxToPart(int robotId,int partId,
      int opFrameId,    int linearMovement,
      double xDistance,double yDistance,
      double zDistance)
```

This function approximates the robot indicated in `robotId` parameter to the location of the `opFrameId` of the `partId` parameter. The actual ToolFrame of the robot will result with the same orientation with the referred operation frame of the part but the position will be the same with an offset of `xDistance`, `yDistance`, `zDistance` value in the axes of the operation frame. Note that usually negative values will be used for approximation. The `linearMovement` parameter has the same meaning that in `alMoveRobot`.

More functions in VREAL documentation

15. Robot Attachment Functions

SEE VREAL documentation

16. Input/Output Functions

This set of functions is designed to control robot input/output signals. Each robot has defined a set of digital outputs, digital inputs, analogical outputs and analogical inputs defined in the constants:

```
NUM_DIGITAL_INPUTS                  NUM_ANALOGICAL_INPUTS
NUM_DIGITAL_OUTPUTS           NUM_ANALOGICAL_OUTPUTS
```

- `alSetDigitalOutput`
  ```
  int alSetDigitalOutput(int robotId,
        int digitalOutputNo)
  ```
  This function sets a digital output of a robot.
- `alResetDigitalOutput`
  ```
  int alResetDigitalOutput(int robotId,
        int digitalOutputNo)
  ```
  This function resets a digital output of a robot.
- `alResetAllDigitalOutput`
  ```
  int alResetAllDigitalOutput(int robotId)
  ```
  This function resets all the digital outputs of a robot.
- `alConnectDigitalInput`
  ```
  int alConnectDigitalInput(int robotId,
        int digitalInputNo,int fromRobotId,
        int fromDigitalOutputNo)
  ```
  This function connects in a digital input of a robot the digital output of another robot.
- `alCheckDigitalInput`
  ```
  int alCheckDigitalInput(int robotId,
        int digitalInputNo,int *digitalStatus)
  ```
  This function obtains in `digitalStatus` the state of a digital input of a robot.

More functions in VREAL documentation

17. Environment Functions

SEE VREAL documentation

18. Robot Operation Functions

This set of functions is designed to allow robot operation, including the interaction with the environment.

- `alActiveTrace`
  ```
  int alActiveTrace(int robotId,int active)
  ```
  This function activates or deactivates the trace of the robot given by the `robotId` parameter. If `active` parameter is `ACTIVE_TRACE`, the trace will be activated but if it is `NO_ACTIVE_TRACE`, the trace will be deactivated. The default value is `NO_ACTIVE_TRACE`, that is, the trace must be activated to be generated. When the trace is active, all locations for active ToolFrame trajectory are stored on the trace according to robot motions. Once a trace has been generated it will be drawn until it is hidden or deleted with one of the next two functions.

  Option: SEE VREAL documentation

More functions in VREAL documentation

- `alPickPart`
  ```
  int alPickPart(int robotId, int partId,
        int opFrameId,double toolStatus,
        int checkOpFrame)
  ```
  This function makes the robot indicated in `robotId` parameter to pick the part indicate in `partId` according to the spatial relation between the active ToolFrame of the robot and the operation frame of the part indicated with `opFrameId`.

  When `checkOpFrame` is:
  - `CHECK_COINCIDENCE` the part will be picked only if the active ToolFrame is coincident (with a tolerance of 10^{-3}) with the operation frame.

- `NO_CHECK_COINCIDENCE` the part will be picked anyway, keeping constant the transformation from the active ToolFrame to the part operation frame.

In any case, the active tool status will be changed to the `toolStatus` parameter.
The effect of the function is that the part will be attached to the robot until the robot is forced to place the part.

Options: SEE VREAL documentation

```
int alPickPart(int robotId,double toolStatus)
```

This function (reduced version of previous function) makes the robot indicated in `robotId` parameter to pick the first part that accomplishes the spatial relation between the active ToolFrame of the robot and any operation frame of the part. In any case, the active tool status will be changed to the `toolStatus` parameter. The effect of the function is that the found part will be attached to the robot until the robot is forced to place the part.

- `alPlacePart`

```
int alPlacePart(int robotId, int partId,
      double toolStatus)
```

This function makes the robot indicated in `robotId` parameter to place the part indicate in `partId`. In addition, the active tool status will be changed to the `toolStatus` parameter. The effect of the function is that the part will be detached from the robot. An error is returned if the specified part is not attached to the specified robot.

```
int alPlacePart(int robotId, double toolStatus)
```

This function (reduced version of previous function) makes the robot indicated in `robotId` parameter to place the part attached to the active tool frame of the robot. In addition, the active tool status will be changed to the `toolStatus` parameter. The effect of the function is that the part will be detached from the robot. An error is returned if there is no part attached to the active tool frame of the specified robot.

19. Functions for Auxiliary List of Figures

SEE VREAL documentation

20. Functions for Display

SEE VREAL documentation

21. Distance Functions

SEE VREAL documentation

22. Video Functions

SEE VREAL documentation

23. Collision Check Functions

SEE VREAL documentation

24. VREAL.INI File

SEE VREAL documentation