



Quick Start: Hands-on #1

ROS.org



Handling ROS Tutorial

Introductory tutorial to ROS and its use for robot in-hand manipulation

Author: Silvia Rodríguez-Jiménez
Carlos III University of Madrid (Spain)

<srjimene@ing.uc3m.es>

Date: October 7th, 2012

Table of Contents

1.	Before starting	3
2.	Package summary	3
3.	Overview.....	3
4.	The database.....	4
5.	Usage	5
5.1.	Launch the local server.....	5
5.2.	General usage of iros_st2_database services.....	6
5.2.1.	Get a list of models by acquisition method.....	6
5.2.2.	Delete a model from the database.....	7
5.2.3.	Download a mesh from the database	7
5.2.4.	Insert a model in the database	7
5.2.5.	Update a model.....	9
6.	More information.....	10

1. Before starting

We provide the ROS nodes for interfacing with a specific SQL database; therefore, this tutorial assumes you have installed:

- A PostgreSQL 9.* server installed and running on a local machine.
- A database which has been restored from the provided backup file.
- The packages `iros_st2_database` and `iros_st2_database_msgs`.

These installation steps are covered in the preparation Hands-on #1 tutorial on the website:

<http://mrl.isr.uc.pt/events/iros2012tutorial/#preparations>

Note: The code **has been tested** using ROS **Electric** and **Ubuntu** 11.10.

2. Package summary

For the Hands-on #1 Session, we provide two packages:

- **`iros_st2_database`**: This package contains class definitions and functions for interfacing with a specific SQL database, containing 3D models of a set of common household objects.
- **`iros_st2_database_msgs`**: This package defines the ROS API for the database and contains ROS message wrappers for some of the data retrieval functions of the `iros_st2_database`.

3. Overview

The package **`iros_st2_database`** allows interfacing with a specific SQL database, containing 3D models of a set of common household objects. Each object in the database has a unique identifier which is generated automatically upon insertion based on a sequence. We use a local database because it can be modified, whereas the database hosted at Willow Garage has remote read-only access.

This package provides service to insert, update, delete and download a model. Furthermore, it allows seeing the list of the models taking into account the acquisition method.

4. The database

The database which will be used during this session can be modified and it has been created using the SQL database from `household_objects_database` package as starting point. For a detailed description of the database itself as a SQL entity, its schema, the data that it contains and the design decisions, see the household objects website:

<http://www.ros.org/wiki/household%20objects>

The database schema is organized by tables, which are cross-referenced using a foreign key. The main difference between the `household_objects_database` and the provided one is the definition of the foreign key. Our database allows the cascaded delete and update (instead of *no action*), it means, whenever rows in the master table are deleted (or updated), the respective rows of the child (referencing) table with a matching foreign key column will get deleted (or updated) as well.

```
SQL pane
-- Table: scaled_model
-- DROP TABLE scaled_model;

CREATE TABLE scaled_model
(
  scaled_model_id integer NOT NULL DEFAULT nextval('model_model_id_seq'::regclass),
  scaled_model_scale double precision NOT NULL, -- The scale of the model, relative
  original_model_id integer NOT NULL, -- The ID of the original version of this model
  CONSTRAINT scaled_model_pkey PRIMARY KEY (scaled_model_id),
  CONSTRAINT scaled_model_original_model_id_fkey FOREIGN KEY (original_model_id)
  REFERENCES original_model (original_model_id) MATCH SIMPLE
  ON UPDATE CASCADE ON DELETE CASCADE
)
```

Figure 1: Example of `scaled_model` table definition.

For each object, the database stores the 3D model of its surface (as a triangular mesh), characteristics, grasp points and links to external data files. When a model is stored, a unique identifier (`original_model_id`) is generated automatically in the `original_model` table. Owing to the fact that all grasps in the database are computed on scaled versions of the original models, each original model identifier has its corresponding scaled model identifier in the `scaled_model` table (Figure 2).

The screenshot shows a database interface with a table list on the left and two table views on the right. The table list includes: acquisition_method, capture_region, dbase_task, dbase_task_type, energy_functions, file_path, file_type, grasp, grasp_source, hand, mesh, model_set, model_source, object_paths, original_model, planning_task, scaled_model, scan, scan_source, task, task_outcome, and variable. The 'original_model' table view shows 230 rows with columns: original_model_id [PK] integer, original_mod double precis, original_mod text[], and original_mod text. The 'scaled_model' table view shows 4 rows with columns: scaled_model_id [PK] integer, scaled_mode double precis, and original_model_id integer.

	original_model_id [PK] integer	original_mod double precis	original_mod text[]	original_mod text
1	9272	1	{reusabl	Stanford
2	9274	1	{glass}	IKEA
3	9276	1	{bowl}	IKEA
4	9277	1	{mug}	IKEA
5	9279	1	{bowl}	IKEA
6	9280	1	{glass}	IKEA
7	9281	1	{glass}	IKEA
8	9283	1	{saucer}	IKEA
9	9284	1	{cup}	IKEA
10	9285	1	{mug}	IKEA

230 rows.

	scaled_model_id [PK] integer	scaled_mode double precis	original_model_id integer
1	18633	1	9272
2	18635	1	9274
3	18637	1	9276
4	18638	1	9277

Figure 2: Detail of the original_model table and scaled_model table, where the identifier of the original model and the identifier of its corresponding scaled model are shown.

5. Usage

5.1. Launch the local server

In order to run all ros services execute the launch file to start the database wrapper node:

```
roslaunch iros_st2_database iros_st2_database_server.launch
```

The ROS wrapper establishes a connection with your database server using the following node parameters, which have defined in the parameter file ../iros_st2_database/config/iros_st2_database_server.yaml:

- /household_objects_database/database_host: the address of the database server

- /household_objects_database/database_port: the port to establish the connection on
- /household_objects_database/database_user: the database username
- /household_objects_database/database_password: the password for the database username
- /household_objects_database/database_name: name of the database to connect to

5.2. General usage of iros_st2_database services

Once started the connection, the wrapper node provides the following ROS services:

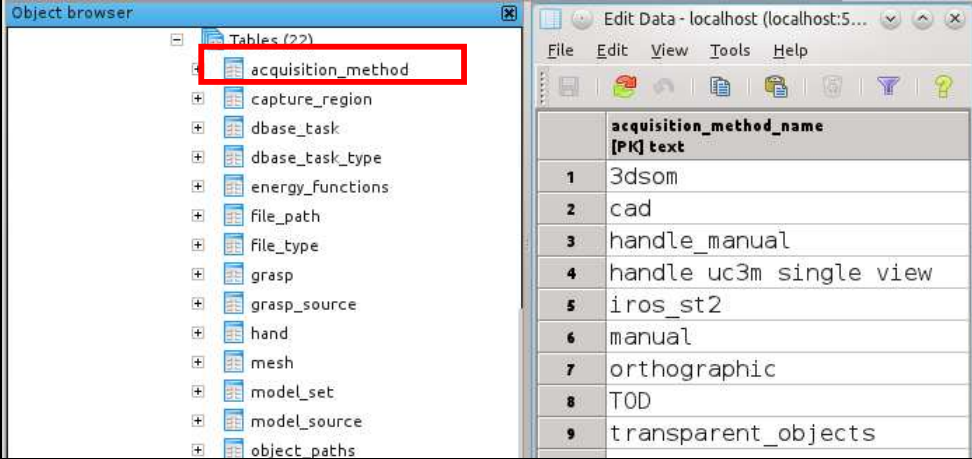
- iros_st2_hdb/get_list_by_acquisition_method: retrieves a list of models available in the database by given the name of the desired acquisition method
- iros_st2_hdb/delete_model: deletes a model from the database
- iros_st2_hdb/download_model: gets the 3D mesh for a given model
- iros_st2_hdb/insert_model: stores a model in the database
- iros_st2_hdb/update_model: updates a model which is stored in the database

5.2.1. Get a list of models by acquisition method

In order to get objects ids, the getModlbyAcquisitionMethod service must be invoked:

- Request: the name of the acquisition method which specifies the search criteria.
- Response: a list of original and scaled objects ids of models available in the database, which have been acquired using the specified method.

The available acquisition methods are stored in the acquisition_method table of the database schema:



	acquisition_method_name [PK] text
1	3dsom
2	cad
3	handle_manual
4	handle_uc3m_single_view
5	iros_st2
6	manual
7	orthographic
8	TOD
9	transparent_objects

Figure 3: Snapshot of the database schema and data of the acquisition_method table.

To test that the service is working, you can use the following command:

```
rosservice call iros_st2_hdb/get_list_by_acquisition_method cad
```

5.2.2. Delete a model from the database

To delete an object the service deleteObjectById should be used.

- Request: the scaled model id of the object to delete from the database.
- Response: the scaled model id of the deleted object.

It could be managed through the following command line:

```
rosservice call iros_st2_hdb/delete_model <scaled_model_id>
```

For example:

```
rosservice call iros_st2_hdb/delete_model 18967
```

5.2.3. Download a mesh from the database

The package provides the option to download mesh(es) from a database and stores them as mesh.stl files in the folder: ../iros_st2_database/database/scaled_model_id.model:

- Request: the scaled model id of the object to download from the database.
- Response: the scaled model id of the downloaded object.

It could be managed through the following command line:

```
rosservice call iros_st2_hdb/download_model <scaled_model_id>
```

For example:

```
rosservice call iros_st2_hdb/download_model 18966
```

5.2.4. Insert a model in the database

The package allows you to add new object models to the database reading the geometry of the model from a .ply file and copying the files to a location specified in the database itself, by the MODEL_ROOT entry in the *variables* table. This variable is used to define a base path that all

paths in the *file_path* table are relative to. Change that to the right path on your machine before trying the insertion of a model:

1. Open pgadmin3 and connect to the server.
2. See the list of the tables: Schemas-->public-->Tables
3. In variable table, right-click: View Data-->View All Rows
4. In the field variable_value text: write your path .../iros_st2_database/database
 - o For example (Figure 4): /home/handle/ros/iros_st2_database/database.

	variable_name text	variable_id [PK] integer	variable_value text
1	MODEL_ROOT	2	/home/handle/ros/iros_st2_database/database

Figure 4: Example of variable table values.

To insert an object the service insertModel should be used. Note that you should make sure that the size of the model is specified in meters and the .ply file contains triangle faces or vertexes:

- Request: description of the object in the following order. In case some information is not available, you should write """.
 - o Name of the object
 - o Name of the acquisition method: the acquisition method used for building a 3D model of this object. One of these values: cad, TOD, manual, orthographic, 3dsom, manual, cad, cad, transparent_objects, iros_st2, handle_uc3m_single_view, handle_manual.
 - o Maker: the maker of that object.
 - o Description: the description of the object.
 - o Barcode: the barcode of the object
 - o geometry_filename: path to the geometry .ply file.
 - o color_image_filename: path to the color .png image file.
- Response: the scaled model id of the stored object.

A unique identifier, *original_model_id* is generated automatically upon insertion based on a sequence. In the model root, a folder called as *original_model_id.model* is created. In this folder, the files related to the object will be copied.

To insert a model in the local household objects database, use the following command line:

```
rosservice call iros_st2_hdb/insert_model <model_name> <acquisition_method>
<maker> <description> <barcode> <geometry_filename> <color_image_filename>
```


For example:

```
rosservice call iros_st2_hdb/insert_model camera iros_st2 "" "" ""  
/home/handle/ros/iros_st2_database/database/examples/camera.model/mesh.ply  
/home/handle/ros/iros_st2_database/database/examples/camera.model/view0000/  
raw/color.png
```

5.2.5. Update a model

To update object data, the service `updateModel` should be used. Updating an object works in the same way as the `insert`, the only difference is that you should provide the id of the object to be updated.

- Request: id of the object to be updated and description of the object in the following order. In case a parameter doesn't need to be updated, you should write `""`.
 - Original model identifier.
 - Name of the object
 - Name of the acquisition method: the acquisition method used for building a 3D model of this object. One of these values: `cad`, `TOD`, `manual`, `orthographic`, `3dsom`, `manual`, `cad`, `cad`, `transparent_objects`, `iros_st2`, `handle_uc3m_single_view`, `handle_manual`.
 - Maker: the maker of that object.
 - Description: the description of the object.
 - Barcode: the barcode of the object
 - `geometry_filename`: path to the geometry `.ply` file.
 - `color_image_filename`: path to the color `.png` image file.
- Response: the original model id of the updated object.

To insert a model in the local household objects database, use the following command line:

```
rosservice call iros_st2_hdb/update_model <original_model_id> <model_name>  
<acquisition_method> <maker> <description> <barcode> <geometry_filename>  
<color_image_filename>
```

For example:

```
rosservice call iros_st2_hdb/update_model 9623 camera iros_st2 unknown "" ""  
"" ""
```

6. More information

Please contact me:

- Silvia Rodríguez-Jiménez <srjimene@ing.uc3m.es>