CMU - Robotics Institute **Manipulation Lab - Simple Hands Project**

Matlab node - ROS - Introduction

<manipulationLab@gmail.edu> October 19, 2011

1 Matlab

Matlab_node is a ROS interface to the MATLAB engine. It allows to execute commands in the MATLAB command line as well as inserting and recover variables from the MATLAB workspace. When executed, matlab_node opens a MATLAB instance in background and without a graphical interface.

It is composed of two ROS packages: matlab_node, a standalone typical ROS package that provides topics and services, and matlab_comm, including the message definitions and a C++ client class to simplify the invocation and communication with matlab_node.

Notes on installation of Matlab engine and how to configure the computer to execute matlab_node can be found at http://simplehands.wikispaces.com/Matlab+Installation.

1.1 matlab_node

(Located at svnroot/code/nodes/matlab/ROS/matlab_node)

Services:

- **matlab_Ping**: Service to ping the matlab interface. It returns true only when the MATLAB engine is opened and all services are ready to serve petitions.

```
---
int64 ret      # Set to 1 (success) or 0 (error)
string msg     # Error description
```

- **matlab_SendCommand**: Service to execute a command in the MATLAB command line. It the command is a function that returns something, we can recover the return value by saving it to a variable in the executed command and recover that variable with the appropriate service.

```
string command    # Command to be executed in the MATLAB command line
---
int64 ret         # Set to 1 (success) or 0 (error)
string msg        # Error description
```

- **matlab_GetArray**: Service to get an array from the MATLAB workspace.

```
string name       # Name of the variable to recover. It should be an array.
---
int64 ncols       # Recovered number of columns of the array.
int64 nrows       # Recovered number of rows of the array.
float64[] data    # Data recovered
int64 ret         # Set to 1 (success) or 0 (error)
string msg        # Error description
```

- **matlab_GetString**: Service to get a string from the MATLAB workspace.

```
string name       # Name of the variable to recover. It should be an array.
---
string data       # Data recovered
int64 ret         # Set to 1 (success) or 0 (error)
string msg        # Error description
```

- **matlab_PutArray**: Service to put an array in the MATLAB workspace.

```
int64 nrows    # Number of rows of the array
int64 ncols    # Number of columns of the array
float64[] data    # Data (row by row)
string name       # Name to give to the array in the MATLAB workspace
---
int64 ret         # Set to 1 (success) or 0 (error)
string msg        # Error description
```

- **matlab_PutString**: Service to put an string in the MATLAB workspace.

```
string name       # Name to give to the string in the MATLAB workspace
string data       # String data
---
int64 ret         # Set to 1 (success) or 0 (error)
string msg        # Error description
```

1.2 matlab_comm

(Located at svnroot/code/nodes/matlab/ROS/matlab_comm)
Matlab_comm is a ROS wrapped C++ class meant to simplify the communication with matlab_node. It contains:

1. Message and Service definitions. When creating a ROS application that uses matlab_node, the application only needs to be dependent on matlab_comm. As a consequence, the application does not need to link to all the matlab engine drivers and libraries. The computer executing the ROS application does not need to have MATLAB installed.

2. C++ class MatlabComm that handles all configuration and initialization required to use matlab_node. As an example, if we want to call the service matlab_Ping, we only need to instantiate a MatlabComm object and call to the member routine matlabPing. Otherwise we would have to include all required header files, subscribe to the matlab_Ping service and format the required message to be sent to the service.

Usage example:

```
...
ros::NodeHandle node;

MatlabComm matlab;           //Create MatlabComm client.
matlab.subscribe(&node);     //Subscribe to all services of matlab_node.
while(!matlab.Ping());       //Wait until MATLAB is ready.
matlab.SendCommand("a=3;");  //Send a command.

...
```

Member routines:

```
class MatlabComm
{
  public:
    MatlabComm();
    MatlabComm(ros::NodeHandle* np);
    ~MatlabComm();

    // Subscription
    void subscribe(ros::NodeHandle* np);

    // Call this before program exits so we don't have double freeing issues
    void shutdown();

    //Client functions to simplify calling Matlab ROS services
    bool Ping();

    //send
    bool sendCommand(const char *command);
    bool sendMat(const char *name, Mat &m);
    bool sendVec(const char *name, Vec &v);
    bool sendValue(const char *name, double v);
    bool sendString(const char *name, char *str);

    //receive
    bool getMat(const char *name, Mat &m);
```

```
    Mat getMat(const char *name);
    bool getVec(const char *name, Vec &v);
    Vec getVec(const char *name);
    double getValue(const char *name);
    bool getValue(const char *name, double *v);
    bool getString(const char *name, char *str, int strLength);
    std::string getString(const char *name);
...
};
```