Code Documentation

Robotic Ultrasound Assistance via Hand-Over-Hand Control

EN 601.656 Computer Integrated Surgery II

Kevin Gilboy
Computer Engineering M.S.E. Student
kevingilboy@jhu.edu

Table of Contents

File Structure

```
$ tree -I
.
+── catkin_build
   +── src
       |── ur_control
       |   |── CMakeLists.txt
       |   |── mainExampleUR.cpp
       |   |── mtsURDerived.cpp
       |   |── mtsURDerived.h
       |   |── Kalman.cpp
       |   +── Kalman.h
       +── CISST
           |── sawRobotiqForceSensor
           |   |── CMakeLists.txt
           |   +── components
           |       |── code
           |       |   +── mtsRobotiqFTSensor.cpp
           |       |── include
           |       |   +── mtsRobotiqFTSensor.h
           |       |── CMakeLists.txt
           |       +── package.xml
           +── sawVarienseForceSensor
               |── CMakeLists.txt
               +── components
                   |── code
                   |   +── mtsVarienseFSensor.cpp
                   |── include
                   |   +── mtsVarienseFSensor.h
                   |── CMakeLists.txt
                   +── package.xml
```

Figure 1: A file tree showing the most relevant files created for this project

mainExampleUR
`./ur-control/`

Description
The main program that creates all CISST components and starts everything.

Public
Functions:

- `int main(int argc, char **argv)`
  Parses command-line arguments then initializes and connects CISST components. Specifically the UR5, Robotiq F/T sensor, and second probe contact force sensor. Once components are created, it starts them all.

---

mtsURDerived
`./ur-control/`

Description
Inherits/Extends the superclass `mtsUniversalRobotScriptRT`, and has added functionality for admittance control.

Public
Functions:

- `mtsURDerived(const std::string &componentName)`
  Constructor to create the component given a string name.

- `mtsURDerived(const mtsTaskContinuousConstructorArg &arg)`
  Constructor to create the component given a generic mtsTask argument

- `void Init(void)`
  Called from the constructor. Creates all state table interfaces necessary for the program and initializes variables.

- `void Configure(const std::string &ipAddr = '', int pose = -1)`
  Configures the IP connection to the UR5 and Robotiq. If pose is -1, the usual hand-over-hand control algorithm is run. Otherwise, the pose indicates which preconfigured pose the robot should home to. There are 32 preconfigured poses stored in the program, which were used during gravity compensation.

- `void Run(void)`
  This function is called repeatedly as part of a typical CISST multitask component. The function collects all necessary data from connected component state tables before calling `ExecuteCommands`.

- `void GetRobotData(void)`
  Fetches the latest robot pose data from the superclass.

- `void GetRobotiqData(void)`
  Fetches the latest Robotiq force data from the Robotiq CISST component's state table.

- `void GetVarienseData(void)`
  Fetches the latest Variense force data from the Variense CISST component's state table.

- `void ExecuteCommands(void)`
  Performs all the necessary steps for gravity compensation, Kalman filtering, and admittance control before setting a velocity goal for the UR5. At the end of this function, `ExecuteCommands()` is called for the superclass.

Private

Variables:

- `InterfaceForceTorque robotiq_ft`
  A custom struct to hold persistent Robotiq force data, relevant constants such as DoF, and handles to read from the Robotiq CISST component's state tables.

- `InterfaceForce variense_f`
  A custom struct to hold persistent Variense force data, relevant constants such as DoF, and handles to read from the Variense CISST component's state tables.

- `Kalman kalman_lin`
  Kalman object meant to filter linear hand forces from the Robotiq. The covariance matrices are set in the `Init(void)` function.

- `Kalman kalman_rot`
  Kalman object meant to filter rotational hand torques from the Robotiq. The covariance matrices are set in the `Init(void)` function.

- `Kalman kalman_probe`
  Kalman object meant to filter linear probe forces from the Variense. The covariance matrices are set in the `Init(void)` function.

Functions:

- `void GravityCompensation(void)`
  Performs gravity compensation using the latest read Robotiq force sensor values. Could be made more generic in the future by using passable force aruments and returning the compensated values.

Kalman

`./ur-control/`

Description

Contains covariences and previous parameters as class variables, has functions necessary to perform Kalman filtering given a measurement vector. Currently hardcoded for a 9 element measurement vector, but should be made more generic using generic CisstVector types in the near future.

Public

Functions:

- `Kalman(void)`
  Constructor that initializes all covariances and persistent state class variables to zero.

- `void SetDt(double dt)`
  Sets the dt of the state matrix A to be the passed parameter `dt`.

- `void SetRCov(vct9 r)`
  Sets the R covariance matrix representing sensor noise to be the passed matrix `r`.

- `void SetQCov(vct9 q)`
  Sets the Q covariance matrix representing action uncertainty to be the passed matrix `q`.

- `void Filter(vct3 x, int samples_since_gt`
  Performs Kalman filtering on 9 values: 3 from the passed x, 3 from the derivative of past x values, and 3 from the second derivative of past x values. The variable `samples_since_gt` signals the number of samples since the last ground truth force reading, which can be used to tweak our covariance confidence.

Private

Variables:

- `double dt`
  Delta time between filter runs.

- `vctFixedSizeMatrix<double,9,9> A`
  State transition matrix.

- `vctFixedSizeMatrix<double,9,9> Qcov`
  Action uncertainty covariance matrix, must be manually set.

- `vctFixedSizeMatrix<double,9,9> Rcov`
  Sensor noise covariance matrix, must be manually set.

- `vctFixedSizeMatrix<double,9,9> P`
  Prediction error covariance matrix, automatically computed.

- `vctFixedSizeMatrix<double,9,9> H`
  The measurement selector matrix, automatically set to the identity.

- `int iter`
  Current iteration number, used to ensure there are no "cold start" issues in differentiating x (e.g. the derivative cannot be computed given one sample).

- `vct9 x[3]`
  Stores three values of x, from times t, t-1, and t-2 respectively.

- `vct9 x_d[2]`
  Stores two values of the derivative of x, from times t and t-1 respectively.

- `vct9 x_dd`
  Stores the second derivative of x.

Functions:

- `vctFixedSizeMatrix<double,9,9> MatrixInv(vctFixedSizeMatrix<double,9,9> M)`
  Inverts a given 9x9 matrix. This is needed since Kalman filtering uses S^{-1} in its optimal gain calculation step.

---

mtsRobotiqFTSensor

`./CISST/sawRobotiqForceSensor`

Description

Extends/Inherits the superclass mtsTaskContinuous, used to fetch TCP data from a Robotiq force sensor.

Public

Functions:

- `mtsRobotiqFTSensor(const std::string & componentName)`
  Constructor that creates the component given the passed component name.

- `void Run(void)`
  An overriden function from the superclass that calls `GetReadings()` and handles the case where the socket disconnects.

- `void Cleanup(void)`
  Closes the Robotiq socket.

- `void Configure(const std::string & ip)`
  An overriden function from the superclass that sets the IP for the Robotiq sensor.

Protected

Functions:

- `void GetReadings(void)`
  Asynchronously listens to the TCP port for force sensor readings, then parses them into the raw data buffer.

- `void Rebias(void)`
  Rebiases the force sensor by sending the official rebias command, as well as calculating a bias vector by averaging several force readings together.

Private

Variables:

- `const static int ROBOTIQ_PORT = 63351`
  The Robotiq TCP/IP port.

- `const static int DOF = 6`
  DoF of the Robotiq.

- `const static int MAX_READINGS_PER_PACKET = 6`
  The maximum number of force readings in a single packet. The number of readings is always between 4 and 6 based on UR5 clock skew and when it decides to send us a packet.

- `osaSocket Socket`
  Socket that communicates with the Robotiq

- `bool IsConnected`
  Boolean flag representing if the connection to the Robotiq was successful

- `double SocketTimeout`
  Timeout value.

- `std::string IP`
  IP address for the Robotiq.
- `int FTDataSeqNum`
  Sequence number for the received force data so that a program asking for force data knows if it is repeated/stale.
- `mtsDoubleVec FTRawData[MAX_READINGS_PER_PACKET]`
  Raw data that the force packet is parsed into.
- `mtsDoubleVec FTData`
  Force data resulting from the raw data buffer.
- `mtsDoubleVec Bias`
  Bias vector since the implicit rebias of the Robotiq only considers its instantaneous force reading. This vector is needed in the case that it rebiases at the peak of noise.

---

mtsVarienseFSensor

`./CISST/sawVarienseForceSensor`

Description

Extends/Inherits the superclass mtsTaskContinuous, used to fetch Serial data from a Variense force sensor.

Public

Functions:

- `mtsVarienseFSensor(const std::string & componentName)`
  Constructor that creates the component given the passed component name.
- `void Run(void)`
  An overriden function from the superclass that calls `GetReadings()` and handles the case where the Serial connection disconnects.
- `void Cleanup(void)`
  Closes the Serial connection.
- `void Configure(const std::string & serialPortName)`
  An overriden function from the superclass that sets the serial port name for the Variense sensor.

Protected

Functions:

- `void GetReadings(void)`
  Asynchronously listens to the serial port for force sensor readings, then parses them into the raw data buffer.
- `void Rebias(void)`
  Rebiases the force sensor by calculating a bias vector through averaging several force readings together.

Private

Variables:

- `const static int DOF = 3`
  DoF of the Robotiq.
- `osaSerialPort SerialPort`
  Serial port that communicates with the Variense
- `bool IsConnected`
  Boolean flag representing if the connection to the Variense was successful
- `int FDataSeqNum`
  Sequence number for the received force data so that a program asking for force data knows if it is repeated/stale.
- `mtsDoubleVec FData`
  Force data read from the sensor.
- `mtsDoubleVec Bias`
  Bias vector.
