A Holistic Data Acquisition Framework for Robotic Surgical Skill Assessment

Description

This is a software system written to combine Dr Malpani's da Vinci recording work with Dr Brown's force-sensing smart taskboard (STB) to create a single unified data-collection system capable of objective surgical skill assessment.

By combining these two systems, we gather synchronized time-series data of both robot manipulator and effector kinematics as well as interaction forces between the robot and surgical workspace. From this data we extract relevant features to predict measures of surgical skill. Finally, these predictions are compared against expert classifications to determine accuracy and reliability of the system.

User Manual

System Setup:

1. Ensure the force sensors cables are plugged in to the DAQ and the usb connected to the computer
2. Connect the peg transfer board to the force sensor
3. Lock the peg transfer board to force sensor in place with screws
4. Place the smart task board in front of the manipulators
5. Power on the da Vinci system via the green button
6. Power on the da Vinci vision tower
7. Press the home button to home the tools
8. Place the desired tools in the da Vinci system
9. Clip the accelerometers onto the manipulator arms
10. Ensure the da Vinci system is connected to the computer
11. Ensure scope angle is set to straight on the da Vinci
12. Sign into the computer
13. Open terminal

Data Collection:

1. Run the following commands in order:

   ```
   cd ~/Desktop/davinci-kinematics
   ./run.sh
   ```

2. Enter an output folder name to begin collecting data
3. Have the user begin the test
4. Enter ctrl-c to end data collection

Data can be analyzed by two programs. One will perform basic statistics and visualize an individual's results.

Data Visualization:

1. Open MATLAB
2. Open DataVisualization2.mlapp
3. Click Select Data button and navigate to the folder containing the trial you would like to visualize
4. Press the debug plot button to verify there is no drift (constant linear graph)
5. Select the number of plots you would like to create
6. Change the streams of data you would like each plot to contain using the add and remove buttons.
7. Change the time over which the series' should be plotted using the slider.
8. Press the plot button to plot the figures.

Known Issues:

- Double check that all cables are connected
- Run MATLAB with software OpenGL

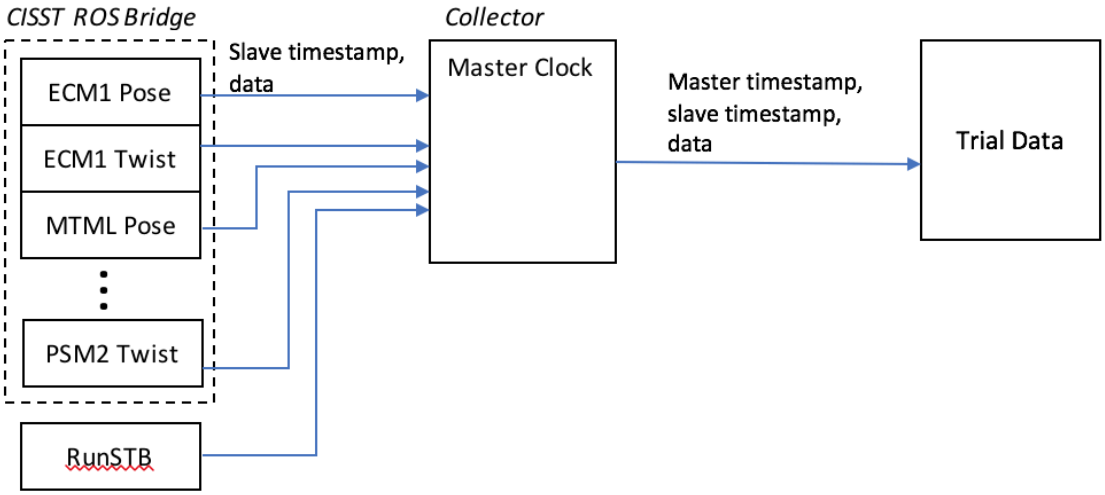
Code Architecture

File Descriptions:

- gztskp/
    - This directory defines a ROS package that contains the executable, collector.
    - The include directory currently holds the decklink library used for video card interfacing (not fully implemented in this version, planned for future versions).
    - The src folder contains:
        - collector.cpp
            - Subscribes to all the data stream topics
            - Defines callbacks that log the data
            - Checks if master clock and slave clocks are drifting.
        - RunSTB.py
            - Main driver for the Dr. Brown side of the system which collects force and acceleration data from the Teensy microcontroller (via USB) and sends the data to a ROS node.
            - This has been edited from its original form to send data a ROS node instead of printing to file.
            - This file depends on all files with the prefix hapticsstb.
        - All files with hapticsstb prefix
            - Contains code from Dr. Brown system that is used by RunSTB.py
            - Please refer to https://git.lcsr.jhu.edu/jbrow262/HAMR_STB andrequest access for more information.
- run.sh

    - Initializes the ROS core, stb ROS publisher node, and the da Vinci ROS publisher node in background tmux sessions.
    - Initializes to the foreground the ROS collector subscriber node.
        - The ROS core is necessary for all ROS packages.
        - The stb ROS publisher node publishes data from RunSTB.py to a ROS topic.
        - The the da Vinci ROS publisher node advertises multiple topics (one for each data stream).
            - This calls the executable isi_console given in the CISST library written by Anton Degeut.
            - For more information on the CISST library please refer to: https://github.com/jhu-cisst
        - The ROS collector node asks for a trial name, subscribes to all topics, timestamps the data on its own master clock and saves the data to file.
- demo_run.sh
    - Does all the same things as run.sh above and forces the trial name to be “demo”, and automatically sends the collected data to calc_stats.py upon completion.
- calc_stats.py
    - Takes a trial folder as a parameter and prints to the console various statistics that describe the trial such as total path length, force integral, max force, total elapsed time, mean velocity.
- VisualizeData2.mlapp
    - MATLAB application to visualize data from a trial.
    - Each button has associated callback function which executes when pressed.
    - Can open in MATLAB to add more components (buttons, sliders, etc.) in design view and add callback functions in code view.
    - Additional variables can be added to the program via application properties.
- README.md
    - Read me for the project. Provides overall description, file details, and directions for running.

Data flow:


CISST ROS Bridge
ECM1 Pose
ECM1 Twist
MTML Pose
PSM2 Twist
RunSTB
Slave timestamp, data
Collector
Master Clock
Master timestamp, slave timestamp, data
Trial Data