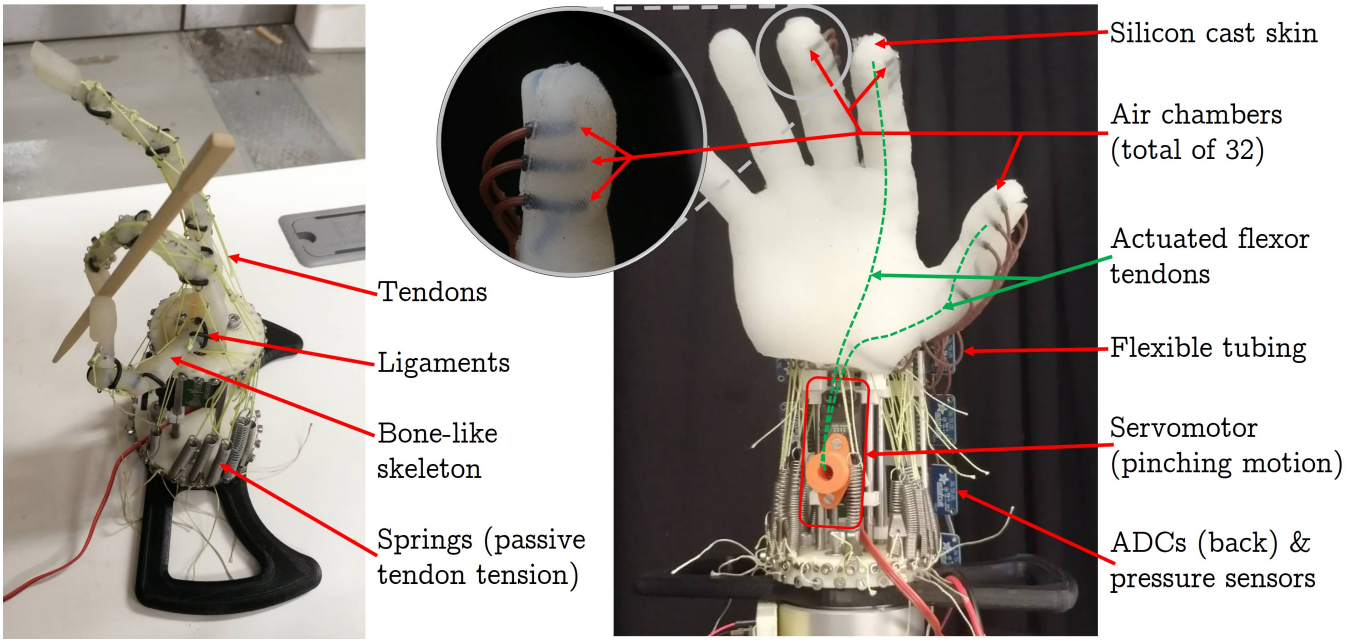
IIB Project: A soft sensorised anthropomorphic hand for improved robotic manipulation



This documentation is provided as a complement to the Master's thesis submitted. It contains a description of the main software used in this project. The software and relevant data has been uploaded to https://github.com/louis-relandeau/IIB-Project-Soft-Sensing which contains a formatted duplicate of this README.

Table of contents:

Main controller

This section describes the controller used for experiments and data collection in this project. It is based off previous work done in the Bio-Inspired Robotics Lab (BIRL), notably PhD student Kieran Gilday.

Contents

1. The Main-controller folder contains ur5_and_hand_controller which is the directory containing all controllers, collected data and function used for plotting. The data, the plotting functions as well as the functions used to collect said data are all named appropriately and briefly commented on in the code.
2. It also contains Arduino, which is all the code used for sensor polling and servomotor control. This code should be uploaded to an Arduino or equivalent following the indicated pin numbering to allow for a controller to communicate with it.

3. This folder also contains ur5_client, the code used on the UR5 client which is required to inrface with the python script. This should be loaded on the UR5 system previously to any connection, see below.

Libraries in use:

- numpy
- matplotlib
- csv
- cv2
- json
- math
- time
- scipy
- random
- pathlib
- _thread
- socket

Description:

Connection is done with the PC running python as a host, the robot will connect to the pc through ethernet(connected on the underside of the robot control box), allowing commmands to be sent over socket. the robot is controlled by calling functions from the kg_robot.py class e.g.:

- `robot=kg_robot.kg_robot()` creates a kg_robot object called robot
- `robot.translatel_rel([0,0,-0.1])` moves the robot in the z direction by -0.1m

In addition to this controller, the PC can connect to the robot 'dashboard server' using the module kg_robot_dashboard.py, where the robot acts as the host. predefined commands can be sent over this connection, such as "load kg_client.urp\n" and "play\n". (more details at UR website)

Connection Setup:

Load Client Program onto robot

- Copy contents of ur5_client folder to a memory stick and plug into robot teach pendant
- Save kg_client.urp to ur5 by loading from the loading tab and pressing 'save as' from the 'file' drop down menu, save in the \programs directory.

Setup ip and ports

- kg_robot:
  - the general connection from kg_client will use the ip address of the computers ethernet connection. for windows open command prompt and enter ipconfig and find the ipv4 address of the ethernet adapter. set self.host in kg_robot.py line 27 and address of var_1:=socket_open('address',port) in kg_client.urp line 8, to this.
  - this connection uses self.port, this is set in python when initialising the kg_robot object, see example in generic_ur5_controller\generic_ur5_controller.py. this must match the port in

var_1:=socket_open('address',port) kg_client.urp line 8. use 30000 or >=30010 (29999 reserved for dashboard, 30001-30004 reserved for data exchange)

- kg_robot_dashboard
  - the dashboard port uses the ip address of the robot, the ip address of the robot is set by returning to the home page of the pedant, selecting 'setup robot' then 'network', selecting static address, inputting the new ip and applying. this is set in python when initialising kg_robot. the port of this connection is 29999 and should not be changed.
  - this address should be set to match the ip of the host computer with the last number different. e.g. if the host is 192.168.1.100 set the robot static address to 192.168.1.x where x!=100
- end effector
  - some support for serial connections, you may need to setup your own communications protocol
  - ee_port: e.g. 'COM20' of any connected end effectors

Getting Started:

- Main loop in Generic_ur5_controller.py, contains examples of using kg_robot and teach mode. The implementation used is in sensing_hand.py.
- Control the robot using the functions defined in kg_robot.py in the ur5 commands section, if the desired robot function doesn't exist and cannot be created through a combination of existing functions, it can be added by modifying kg_robot.py and kg_client.urp. Complete capabilities are described in ur5 script api.
- Create specialised robot fns and attach in same format as teach_mode.py by adding an init to kg_robot.py, e.g. complex move sequences, cameras, calibrations and robot workspaces.
- Use waypoint.py for global robot poses, joints and tool centre points (tcp).

Neural Networks

This section relies on a set of iPython notebook based on a tutorial from machinelearningmastery. The Neural-networks contains the code used for orientation predictions using an ANN and LSTM network, the data used to train and test these networks, and the obtained results. Folders containing results for each of these implementations are labelled appropriately.

Libraries in use:

- Sklearn
- Keras
- TensorFlow
- Numpy
- Pandas
- Matplotlib
- Csv

Getting Started:

The folders orientation_prediction_ANN and orientation_prediction_LSTM each contain a .iynb file which describe the entire training process and results for a certain type of network. A comparison of both results is included in ANN-LSTM-comparison.ipynb.

The implementations follow the following structure:

1. Importing the data from a .csv file
2. Processing the raw data to extract suitable dataset and format it appropriately for use in neural networks
3. Splitting the data in training and testing datasets with shuffle if required
4. Define model topology
5. Perform training and plot loss in real time
6. Print an example of prediction from the testing dataset

These steps can be followed to understand the methods used and replicate these techniques for similar problems.