Semester project
School of Computer and Communication Sciences

Learning ergonomic human-robot interaction

Spring 2019 **LASA** - EPFL

By :
Antoine Laurens

Supervisor :
Baptiste Busch
Mahdi Khoramshahi

Professor:
Aude Billard
Wulfram Gerstner

Contents

Chapter 1

Introduction

Humans perform many different tasks on a daily basis. For example preparing your food in the morning walking down the street. Also many of these tasks are work related such as sitting at a desk, performing manual work on a construction site (Fig. 1a) or a dentist examining his patients. Each of these tasks requires that we adopt a certain posture to perform the task efficiently. These postures can be held for extended periods of time and studies have shown that prolonged exposure to a bad posture can lead to pain and injuries [1]. These are referred to as musculoskeletal disorders (MSDs) and are a problem in many countries [2]. For example, dental students develop such MSD during their practical courses because of the posture they adopt when examining patients [3]. In the context of this project we ask ourselves if robots can help humans with their posture?

To estimate the comfort and safety of these postures, methods such as the Rapid Entire Body Assessment (REBA) have been developed [4]. This method allows to estimate the comfort of a static pose by assigning a score to each joint and summing each individual score. By combining this evaluation with real time tracking of the joint configuration of a user, one can imagine several different applications. For example, a large amount of data can be collected to create a model used to indicate to the user if he is in a bad posture such as those described in [5]. Could this method be used to create a cost function that a robot could then minimize?

When two humans are performing a task together each of them will move in reaction to the other. For example when carrying a large object, both individuals will ensure to walk at the same speed. Therefore, if one starts to slow down, the other adapts his movements accordingly. However, when the work is shared between a human and robot, the workload is not balanced equally, the robot must assist the human as much as possible. For example, if a robot is helping a human carry a heavy load, it must support most of the weight.

The goal of this project is to use this human robot interaction to explore the possibility of having a robot act on the environment of the user allowing him to adopt a more ergonomic and comfortable posture to execute a task. For example, one can imagine the desk and the chair of an office worker moving to put the employee in a safer position. Or the chair at the dentist moving so the doctor doesn't have to bend over excessively. Methods are proposed to reduce the effort of the user in a human-robot interaction by having a cost function measure the fatigue of the human and adapting the behavior of the robot to help him more when he tires [6]. Here, the interest is more in posture than in muscle fatigue, nonetheless, cost function and adaptation are used here. In [7] the user is interacting with a robot to perform a task and the robot is able to reach an optimal position that allows the user to perform the task efficiently while also adopting a safe posture. The limitation however in the previously described project is that the position of the robot is computed off-line. The robot is then brought to the computed position and maintains it with respect to the human. This project focuses on real time postural improvement of the human through robotic motion.

To illustrate the approach developed in the project, the example of optimizing the posture of a user screwing on a box held by the robot is chosen (Fig. 2). This simple task serves as proof of concept for the proposed method. The idea is to create a model linking the cost of a posture to the joint configuration. The robot then minimizes this cost by moving the box and allows the human to adopt a more comfortable position to perform the task. To achieve this, the robot's motion planner takes features from the joint configuration as input and, using parameters that are previously optimized, outputs a velocity command for the robot to follow.

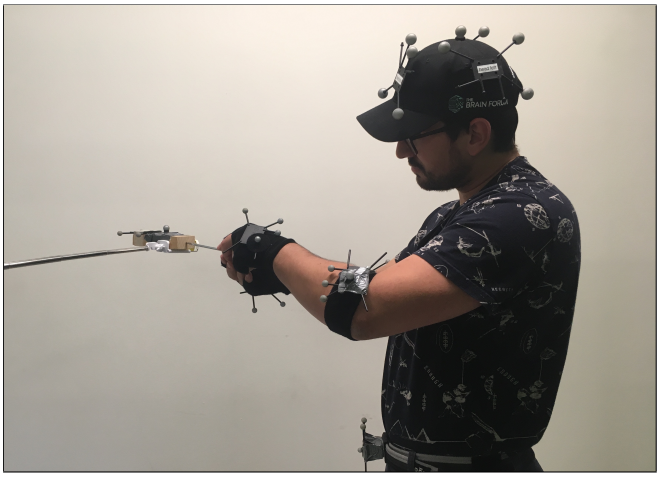
The first part of this report describes different tools that are used in the project such as Gaussian Mixture Models, Support Vector Machine, the REBA method as well as different means for data collection. The proposed method for postural improvement is then described. The controller used as well as the algorithm to find its parameters are explained. A toy example is presented to assess the feasibility of the method. Then, the experimental setup used for the real test case is exposed. Finally the results are discussed and a conclusion to the project is given.

(a)

(b)

Figure 1: **(a)** Example of a construction worker performing a task using an unsafe posture [8]. **(b)** Example of a human working in interaction with a robot [9].

(a)



(b)

Figure 2: **(a)** Position of the worker before **(a)** and after **(b)** postural improvement when performing the task of screwing on an object held by the robot (here replaced by a selfie-stick). The robot must bring the user from an uncomfortable position to a comfortable one.

Chapter 2

Background

2.1 Gaussian Mixture Regression and its derivative

2.1.1 Gaussian Mixture Model (GMM)

GMM [10] is an unsupervised learning method to fit a given number of Gaussian over some data. The Expected Maximization (EM) algorithm optimizes the prior, the mean and covariance matrix of each Gaussian to increase the likelihood of the model.

When using GMM, one can compute the probability that an unseens data point $\xi \in \mathbb{R}^n$ was generated by the mixture of Gaussian:

$$P(\xi) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\xi|\mu_k, \Sigma_k) \tag{2.1}$$

$$= \sum_{k=1}^{K} \frac{\pi_k}{\sqrt{(2\pi)^n|\Sigma_k|}} \exp\left(-\frac{1}{2}(\xi - \mu_k)^\top (\Sigma_k)^{-1}(\xi - \mu_k)\right) \tag{2.2}$$

where K is the number of Gaussians and $\pi_k \in \mathbb{R}$, $\mu_k \in \mathbb{R}^n$ and $\Sigma_k \in \mathbb{R}^{n\times n}$ are respectively the prior probability, the mean and the covariance matrix of the k^{th} Gaussian.

2.1.2 Gaussian Mixture Regression (GMR)

Once the data has been fitted, incomplete data can be estimated using GMR [11]. The following decomposition of the data point, mean vectors and covariance matrices is given:

$$\xi = \begin{bmatrix} \xi^I \\ \xi^O \end{bmatrix}; \quad \mu_k = \begin{bmatrix} \mu_k^I \\ \mu_k^O \end{bmatrix}; \quad \Sigma_k = \begin{bmatrix} \Sigma_k^I & \Sigma_k^{IO} \\ \Sigma_k^{OI} & \Sigma_k^O \end{bmatrix} \tag{2.3}$$

where I and O respectively represent the input and output components. Equation 2.1 now gives the joint probability distribution $P(\xi^I, \xi^O)$. Conditioning on the known components will give a new distribution for the output terms:

$$P(\xi^O|\xi^I) = \sum_{k=1}^{K} h_k \mathcal{N}(\xi^O|\hat{\mu}_k^O, \Sigma_k^O) \tag{2.4}$$

with

$$\hat{\mu}_k^O = \mu_k^O + \Sigma_k^{OI}(\Sigma_k^I)^{-1}(\xi^I - \mu_k^I)$$

$$\hat{\Sigma}_k^O = \Sigma_k^{IO} - \Sigma_k^{OI}(\Sigma_k^I)^{-1}\Sigma_k^{OI}$$

$$h_k = \frac{\pi_k \mathcal{N}(\xi^I|\mu_k^I, \Sigma_k^I)}{\sum_{i=1}^{K} \pi_i \mathcal{N}(\xi^I|\mu_i^I, \Sigma_i^I)}$$

The new distribution is also a GMM of lower dimension. The output can also be expressed as a single Gaussian with mean $\hat{\mu}^O$ and covariance matrix $\hat{\Sigma}^O$:

$$\hat{\mu}^O = \sum_{k=1}^{K} h_k \hat{\mu}_k^O \tag{2.5}$$

$$\hat{\Sigma}^O = \sum_{k=1}^{K} h_k (\hat{\Sigma}_k^O + \hat{\mu}_k^O (\hat{\mu}_k^O)^\top) - \hat{\mu}^O (\hat{\mu}^O)^\top \tag{2.6}$$

Fig.3 shows an example of such a regression.

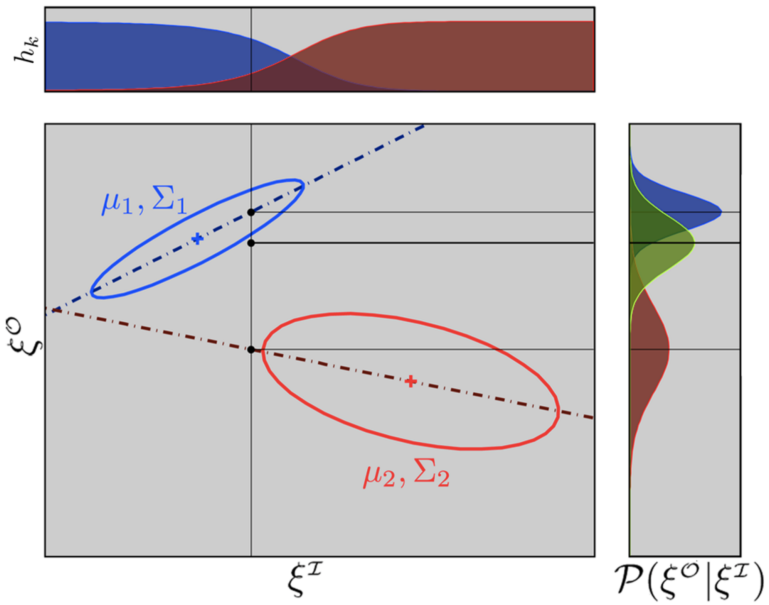


Figure 3: Example of GMR. The Guassians fit 2 dimensional data. The conditioning is done on the x-axis. On the right the result of the regression is showed. The red and green Gaussian represent the new GMM associated to the ouput data and the green one represents the combined result. [12]

2.1.3 Derivative of a GMR

When approximating a function using GMR, one might also be interested in obtaining the derivative of this approximation. More particularly, the derivative of $\hat{\mu}^O$ (Eq. 2.5) is computed as follows:

$$\frac{\partial \hat{\mu}^O}{\partial \xi^I} = \frac{\partial}{\partial \xi^I} \sum_{k=1}^{K} h_k [\mu_k^O + \Sigma_k^{OI} (\Sigma_k^I)^{-1} (\xi^I - \mu_k^I)] \tag{2.7}$$

$$= \sum_{k=1}^{K} h_k \Sigma_k^{OI} (\Sigma_k^I)^{-1} + \hat{\mu}_k^O \frac{\partial h_k}{\partial \xi^I} \tag{2.8}$$

with

$$\frac{\partial h_k}{\partial \xi^I} = h_k \left[A_k - \frac{\sum_{j=1}^{K} A_j \pi_j \mathcal{N}(\xi^I | \mu_j^I, \Sigma_j^I)}{\sum_{i=1}^{K} \pi_i \mathcal{N}(\xi^I | \mu_i^I, \Sigma_i^I)} \right]$$

where $A_k = -\left[(\xi^I - \mu_k^I)(\Sigma_k^I)^{-1} \right]^\top$. The term $\Sigma_k^{OI}(\Sigma_k^I)^{-1}$ is the slope of the local linear fit around the Gaussian (the dashed lines in Fig.3).

Fig.4 shows an example of performing regression and computing the derivative on a sampled cosine function.

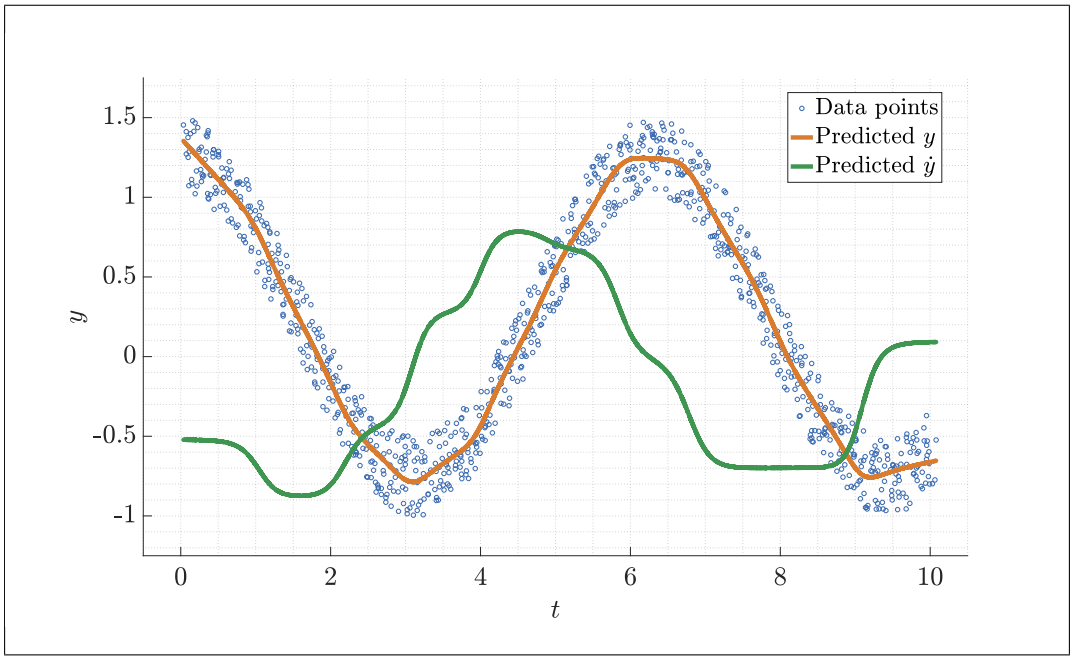


Figure 4: Example of GMR and derivative of GMR on noisy cosine data. The model successfully extracts the cosine function (in orange) as well as its derivative (in green).

2.2 Evaluating the cost of a posture: the REBA method

The REBA method [4] allows to rapidly and systematically evaluate the cost of a posture when performing a work related task. This cost is linked to the risk of a posture to perform the task, the lower the cost the safer the posture. It is important to know if a worker is executing a task in a dangerous position because repeated exposure to such posture leads to injuries. It is therefore important to correct such postures early on and not when pain and discomfort appear.

When evaluating a position using the REBA method, an evaluator assigns a score to each of the following body parts: legs, knees, back, trunk, neck, shoulders, elbow and wrist. The individual scores are determined by looking at the different lookup tables in Fig.5. For example Table A determines the "trunk score" whereas Table B is used for the "lower arm score". The final cost is found using the Table C and adding an "activity score" to it. The score goes from 1 indicating a very safe posture all the way up to 12, a dangerous position that must be changed as soon as possible.

In Fig.5 the evaluation sheet of the REBA method is showed. Following this guide-sheet, it is possible to evaluate the score of the worker in Fig.1a. The awkward position of his should results in a high score for that body part indicating that the position is unsafe and a change should be made to prevent injuries.

This method however is not suitable "as is" for this project. Indeed, for every posture a discrete score is assigned and there is no linear relationship between the final score and individual scores of each body parts. Otherwise the score could directly be computed by using that linear combination. There is also no derivative with respect to the joints. As explained in the next chapter, for the robot to be able to bring the human to a more ergonomic position, the derivative of the cost with respect to the joints must be computable. To overcome this problem the use of Gaussian Mixture Models as well as Support Vector Machine is explored.

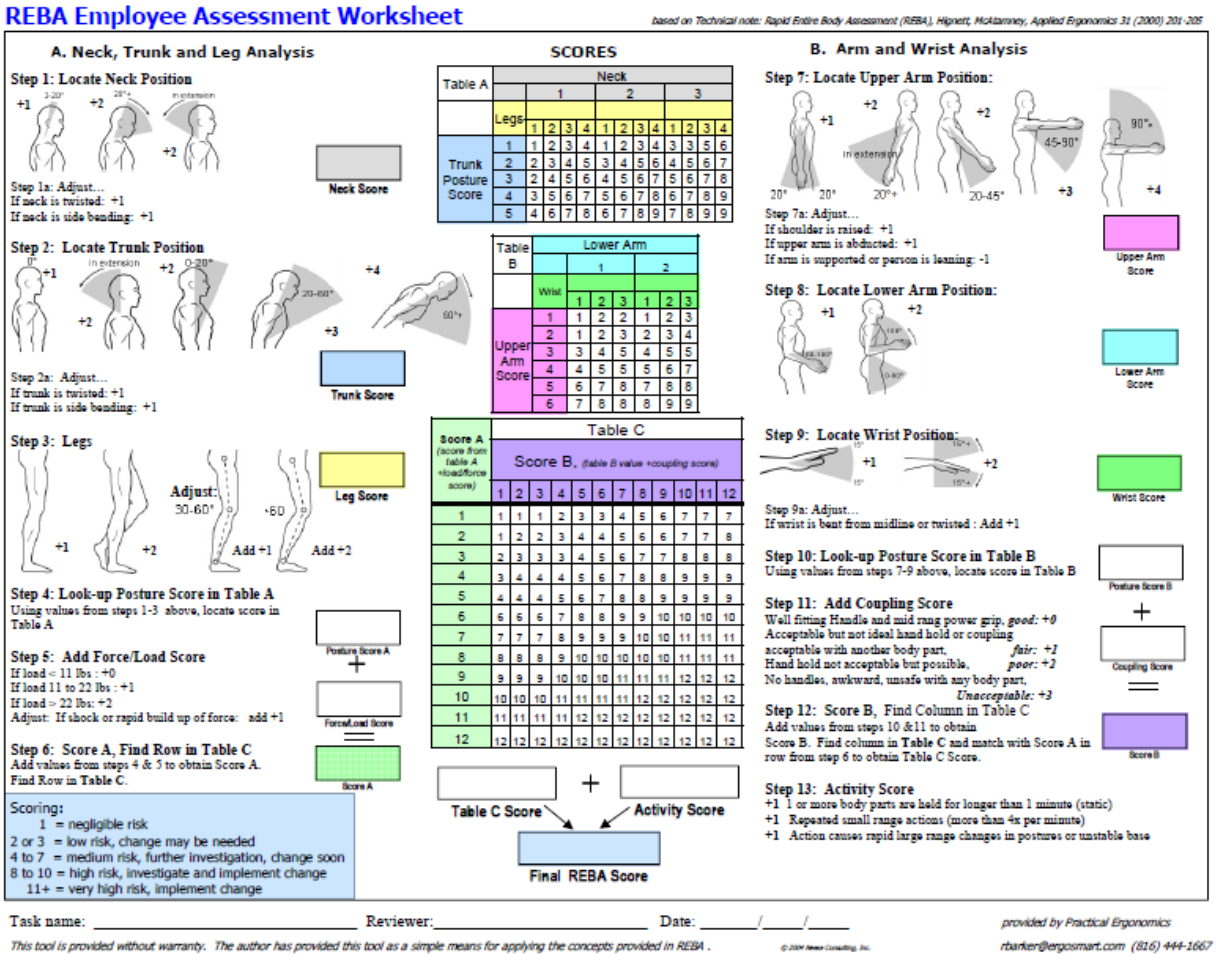


Figure 5: REBA table for posture evaluation [13]. Each body part is evaluated separately, for example the top left corner of the sheet evaluates the neck. Each score is then added and the REBA score is obtained at the bottom of the sheet.

2.3 Data collection

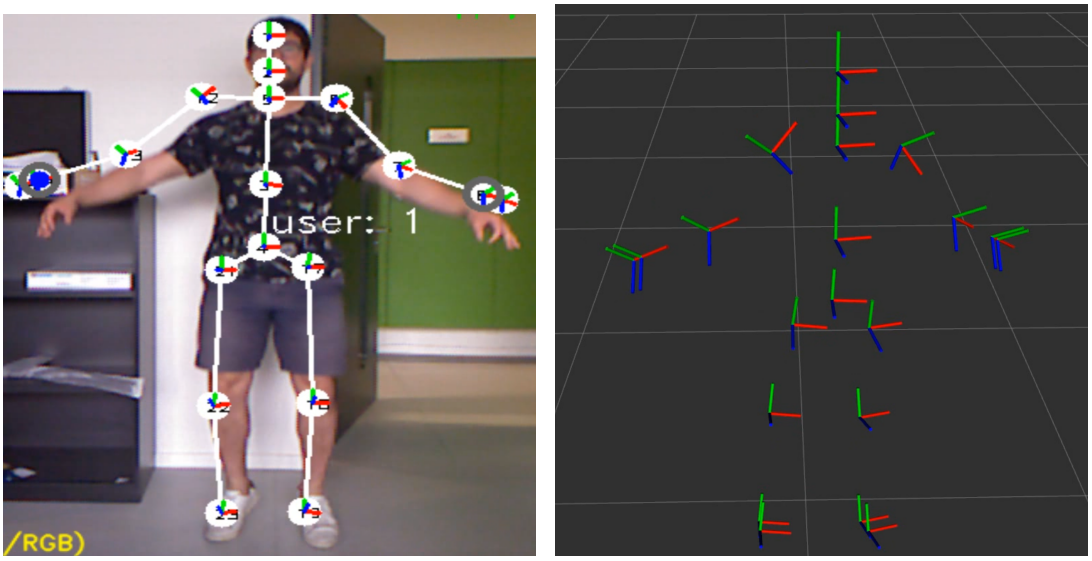
2.3.1 Kinect

The Kinect is a system that was created by Microsoft to be used as a controller for the Xbox360. However, the cheap price, ease of installation and non-invasive measurement system makes the Kinect of tool of choice for research and many independent developers around the world [14]. The Kinect works by projecting an array of infrared light to the scene in front of it and measuring of depth array of each point. This depth array can then be used to extract different features.

Here, Nuitrack Full Body Skeletal Tracking Software is used [15]. It extracts the skeleton of a human placed in front of the camera as illustrated in Fig.6a. A ROS package is written so that the data from Nuitrack can be extracted and each joint is send to ROS as a tf, a ROS object representing 6D transformations. A visualization of the collected tfs can be observed in Fig.6b.

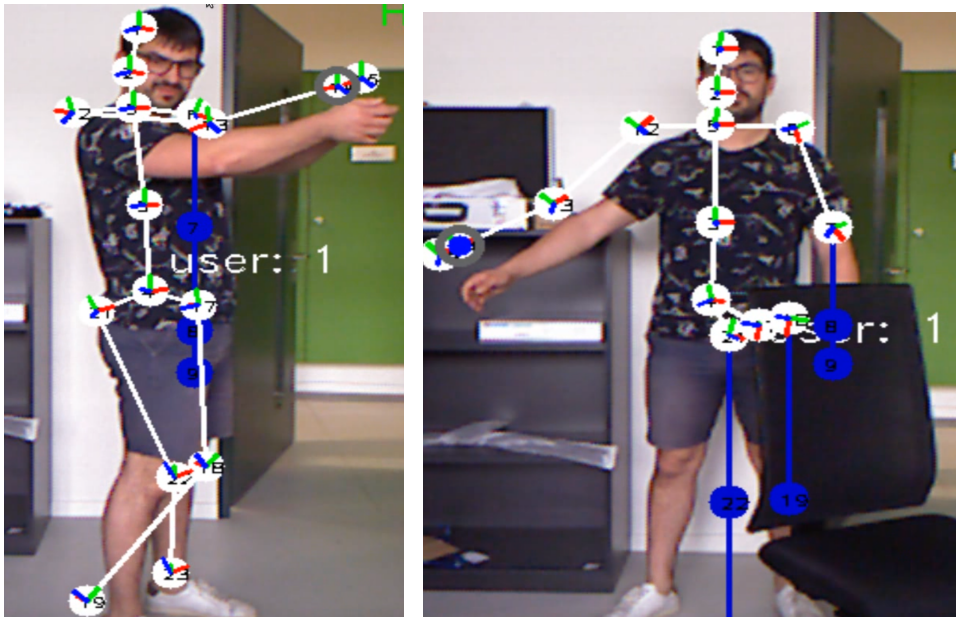
The Kinect sensor is used to generate a scaled 32DOF model of the human by calculating the lengths of each segments. However the tracking is not ideal as the human must be facing the camera and no obstacles must be

between the user and the camera. If this is not the case, the system is not able to detect all the joints as illustrated in Fig.7. In fact, any posture where each body part is not clearly visible by the camera causes issues during the detection. This is a problem both during data collection and when the human is interacting with the actual robot. Therefore, Optitrack is used as the final tracking system, calibrated with the previously generated human model. For additional details on the method see [7].



(a) (b)

Figure 6: **(a)** Skeleton detection using Nuitrack and the Kinect. **(b)** Visualization of the joints detected by Nuitrack in ROS. [9].



(a) (b)

Figure 7: Kinect unable to detect the whole skeleton when **(a)** not facing the camera. **(b)** there is an object blocking.

2.3.2 Optitrack

The Optitrack system allows to track markers using a set of cameras laid around the scene. This means that multiple cameras track the markers and that even if some cameras are blocked the tracking system still works. This makes the Optitrack system more robust than the Kinect and is therefore better suited for this project.

To track every required joint eight markers are used and placed around the body as well as one marker used to track the object on which the human is screwing (Fig.8):

- head
- shoulder center
- elbows
- hands
- belt (used as the base frame)

Figure 8: Marker positioning when collecting the data with Optitrack. 8 markers are placed on the body to measure the joints positions and one is placed on the object the human is operating on. Forward Kinematics is then used to extract the joint configuration

Chapter 3

Proposed method

3.1 Optimizing the human posture

3.1.1 The control loop

In this chapter, the mathematical formulation for the improvement of the human posture is presented.

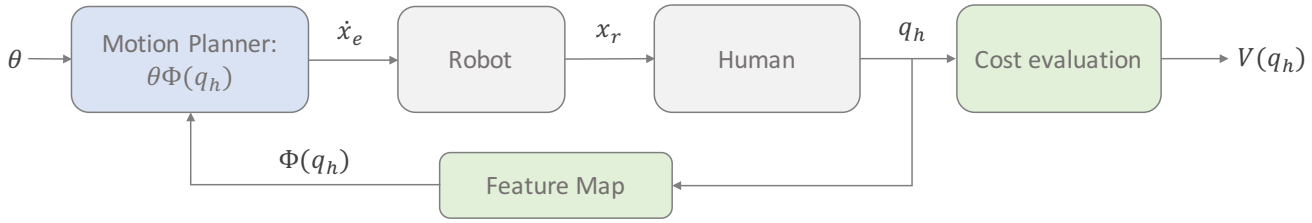


Figure 9: Control loop used to improve the cost $V(q_h)$ of the human joint configuration q_h. A feature map Φ extracts features from the current q_h. A motion planner then computes an estimated velocity $\dot{x}_e = \theta\Phi(q_h)$ using a set of parameters θ. The robot finally moves and the human follows which leads to a q_h with a reduced cost.

Fig.9 shows a diagram of the control loop detailed below.

We call $q_h \in \mathbb{R}^N$ the joint configuration of the human. The goal is:

$$\min_{q_h} \quad V(q_h)$$

where $V(q_h) \in \mathbb{R}$ is the cost associated to the current joint configuration. This minimization is done through robotic motion. The joint configuration q_h is passed through a feature map $\Phi : \mathbb{R}^N \to \mathbb{R}^M$ that extracts meaningful features that are then sent to the controller.

The controller then computes a velocity command to send to the robot:

$$\dot{x}_e = \theta\Phi(q_h) \tag{3.1}$$

where $\dot{x}_e \in \mathbb{R}^P$ is the estimated velocity command that is sent to the robot and $\theta \in \mathbb{R}^{P\times M}$ is the controller parameters. θ is optimized to produce a controller able to improve the human's posture.

The robot uses an impedance-based velocity controller to allow the human to safely interact with it [16]. When receiving this command the robot moves to a new pose $x_r \in \mathbb{R}^6$. This in turn forces the human to move to continue performing the task properly. This new posture q_h has a lower cost $V(q_h)$.

The problem is now to find the parameters θ of the controller such that the final cost of the human posture is minimized:

$$\min_{\theta} \quad V(q_h)$$

To perform this minimization the problem is divided into 2 sub-problems:

- The first problem is finding the desired velocity command $\dot{x}_d$ that must be sent to the robot to make the human human move in the correct way.

- Once the desired velocity command is known, θ must be tuned to reduce the difference between the estimated velocity and the desired one: $\dot{x}_e - \dot{x}_d$

To find the $\dot{x}_e$ that brings the human into a safer configuration, gradient descent is applied to the cost $V(q_h)$:

$$\dot{x}_d = -\epsilon \frac{\partial V(q_h)}{\partial x_r} \tag{3.2}$$

$$= -\epsilon \frac{\partial V(q_h)}{\partial q_h} \frac{\partial q_h}{\partial x_r} \tag{3.3}$$

with $\epsilon \in \mathbb{R}$ a scaling factor. The first term, $\frac{\partial V(q_h)}{\partial q_h}$, is the derivative of the cost with respect to the joint configuration of the human. It explains the evolution of the cost when the human changes his posture. The second term, $\frac{\partial q_h}{\partial x_r}$, is the partial derivative of the human joint configuration with respect to the robot. This term describes how the human moves when the robot moves. This component is highly dependent on the task, indeed the way one moves depends on what he is trying to do. You do not move in the same way when you are sitting at desk and when you are screwing on a wall.

To find the correct $\dot{x}_d$ two models must be created. A first model for the joint configuration and its cost. For this, the REBA method [4] is used (see section 2.2) to attribute a cost for a given joint configuration. The second model describes q_h in relation to x_r. To obtain this model, a user is recorded while performing a task.

Once the correct desired velocity is computed the parameter θ can be optimized using a gradient based method. Given:

$$J = \frac{1}{2} \|e\|^2 \tag{3.4}$$

where $e = \dot{x}_e - \dot{x}_d$ is the error vector between the estimated ans the desired velocity. The parameter θ can be changed using gradient descent:

$$\dot{\theta} = -\lambda \frac{\partial J}{\partial \theta} \tag{3.5}$$

$$= -\lambda \frac{1}{2} \frac{\partial e^\top e}{\partial \theta} \tag{3.6}$$

$$= -\lambda e \frac{\partial e}{\partial \theta} \tag{3.7}$$

$$= -\lambda e \Phi(q_h)^\top \tag{3.8}$$

where $\lambda \in \mathbb{R}$ is scaling factor and 3.8 is obtained using 3.1. With this update rule, θ can be trained using the following algorithm:

Algorithm 1: Optimization of controller parameter θ

Result: Optimized θ

initialization: $\theta_0 = 0_{P \times N}$;

while $\|\theta_{t+1} - \theta_t\|_\infty > threshold$ **do**

  q_h = random joint configuration;

  $\dot{x}_e = \theta \Phi(q_h)$;

  $\dot{x}_d = -\epsilon \frac{\partial V(q_h)}{\partial q_h} \frac{\partial q_h}{\partial x_r}$;

  $e = \dot{x}_e - \dot{x}_d$;

  $\Delta\theta = -\lambda e \Phi(q_h)^T$;

  $\theta_{t+1} = \theta_t + \Delta\theta$

end

3.2 Using GMM and GMR in this project

In this project, GMM and GMR are used to give an evaluation of the cost $V(q_h)$ as well as obtaining the desired velocity $\dot{x}_d$.

The cost however is only available as a look-up table and not as a smooth differentiable function that can be used for control purposes. The only possibility is to sample from it and then estimate it. This is done by generating a dataset with random joint configurations and the associated cost. On top of these samples a GMM can then be fitted. This allows to obtain the joint distribution $\mathcal{P}(V(q_h), q_h)$ that serves as an approximation of the cost function. Using GMR, the cost $\tilde{V}(q_h)$ of a new unseen joint configuration is predicted by conditioning on q_h and taking the expected value:

$$\tilde{V}(q_h) = \mathbb{E}[\mathcal{P}(V(q_h) \mid q_h)]$$

Fig.10 shows a simple example of the method with $q_h \in \mathbb{R}$ defined over $[-6, 6]$. This example shows that, on a simple system, the method can be used and it provides satisfactory results given a sufficient amount of data.

Following this, a second model must be generated to have the joint probability of q_h, the position of the robot x_r and the cost $V(q_h)$. To obtain this model a human performs this task while measurements are taken (section 2.3). Each recorded data point is composed of a joint configuration q_h associated to a robot position x_r. To each of these a predicted cost $\tilde{V}(q_h)$ is associated using the 1^{st} model. Then a 2^{nd} GMM is fitted on this other dataset.

Using these two models $\dot{x}_d$ can now be determined by using equations 3.3 and 2.8. Finally using the previously described algorithm θ can be trained to produce a controller able to bring the system to a minimum. Fig.11 shows and example of a controller obtained through such adaptation using the system presented in Fig.10. The computed controller is able to bring the system into a local optimum.

Nevertheless, other regression methods can also be used such as Support Vector Regression based on Support Vector Machine.

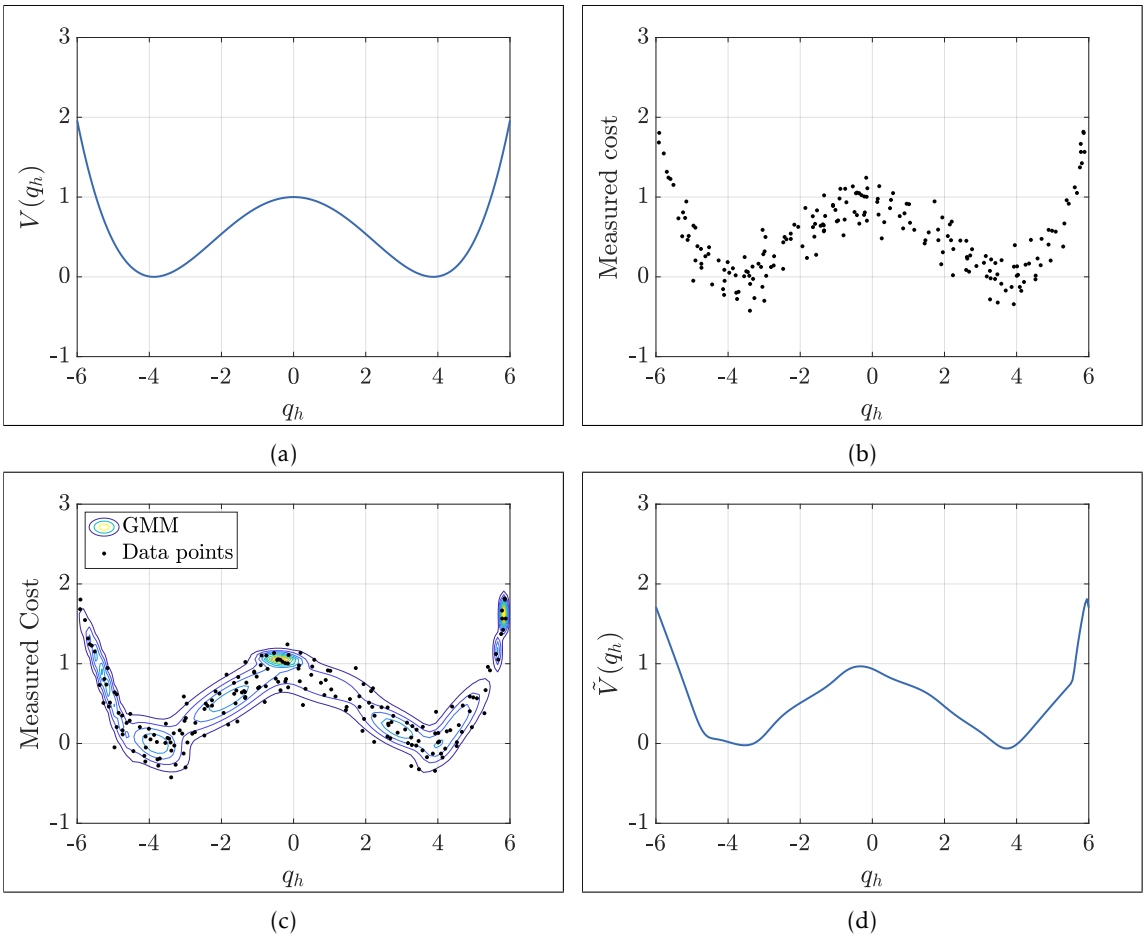


Figure 10: **(a)** Cost function of the system : $V(q_h) = \frac{q_h^4 - 30q_h^2 + 225}{225}$ with $q_h \in [-6;6]$ **(b)** 200 data points obtained by sampling the cost function and adding noise to the result **(c)** GMM model fitted on the noisy data to approximate the cost function **(d)** Result of applying GMR to the model in **c** to predict the cost at every point.

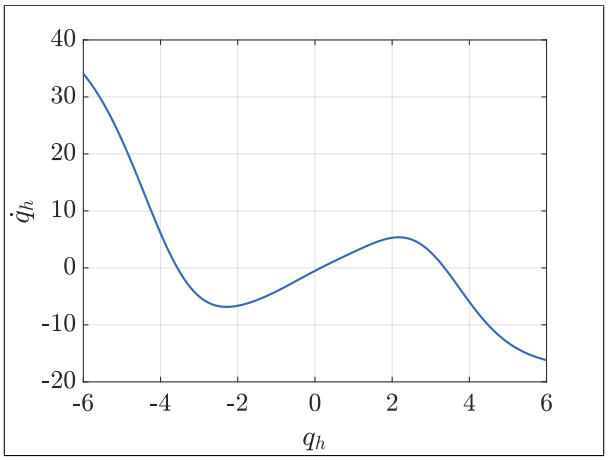


Figure 11: Output of the optimized controller $\dot{x}_d = \theta^\top \Phi(q_h)$ where $\dot{q_h} = \dot{x}_d$ and the feature map $\Phi(q_h) \in \mathbb{R}^{10}$ is 10 RBFs placed evenly in $[-6;6]$.

3.3 Experimental setup

Here, the goal is to implement the method described in the previous chapter on a real task. The chosen task is illustrated in Fig.2. The human must screw on a box and the robot must orient and place the box such that the posture adopted by the human is the most ergonomic possible.

3.3.1 Procedure to measure the data

It is necessary to record data of the human performing the task to understand the human task-specific forward Inverse kinematics in this specific context. The setup for recording the task is shown in Fig.12.

Also, the measured data corresponds to the joints spatial coordinates with respect to a base frame. Here however it is the joint configuration that we are interested in (such the angular configuration of the elbow). Therefore, a package is used to perform the inverse kinematics of the human body and obtain the desired data [17].

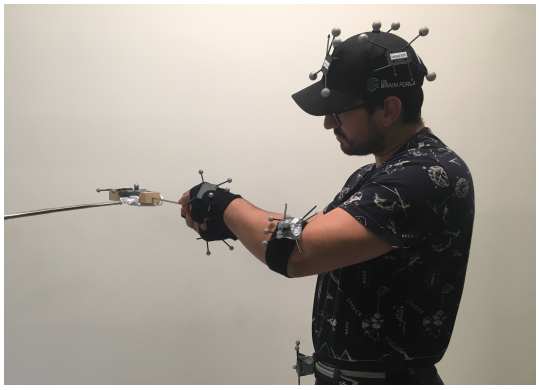


Figure 12: Setup for the data collection. The human performing the task is strapped with multiple markers. Another person is holding a mock object that the user must screw on. This object is also tracked when recording the data. The object is held using a stick to prevent blocking the vision of the object.

Once the markers have been placed as shown in Fig.8 data can start being collected. The procedure to collect the data is the following:

1. Launch `roscore`
2. Launch the Optitrack tracking system using:
   `roslaunch mocap_optitrack epfl_optitrack.launch`
   This ros package can be found here [18]. Make sure that the file `epfl.yaml` has been properly modified so that the markers are correctly tracked and published.
3. Launch the human_movit docker using the scripts `build.sh` and `run.sh` the docker is found here [17]. Do a `catkin_make` when inside the docker.
4. **Inside** the docker create a model of the human while he is in T-pose:
   `roslaunch human_movit_config generate_model.launch`
5. **Inside** the docker calibrate the model while the user is in T-pose:
   `roslaunch human_movit_config calibrate_model.launch`
6. **Inside** the docker launch the tracking of the human:
   `roslaunch human_movit_config human_tracker.launch`
7. **Outside** the docker start collecting the data:
   `rosrun human_tf_pub optitrack_recorder`

Chapter 4

Results

In this chapter the results of the data analysis and simulation of the method are detailed. The data used in this chapter is a dataset that was generated randomly. For each cost value from one to ten, 1'000 joint configurations are generated resulting in a dataset with 10'000 points each compose of 20 joints.

4.1 Using GMM-GMR

4.1.1 Evaluating the cost of an unseen posture

To determine the number of Gaussians needed to fit the model the Bayesian Information Criterion is used [19]. The result of the BIC evaluation is shown if Fig.13. Therefore, 24 Guassians are chosen to model the data.

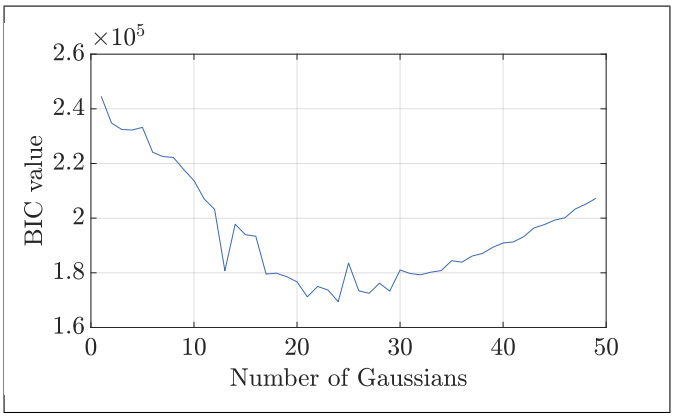
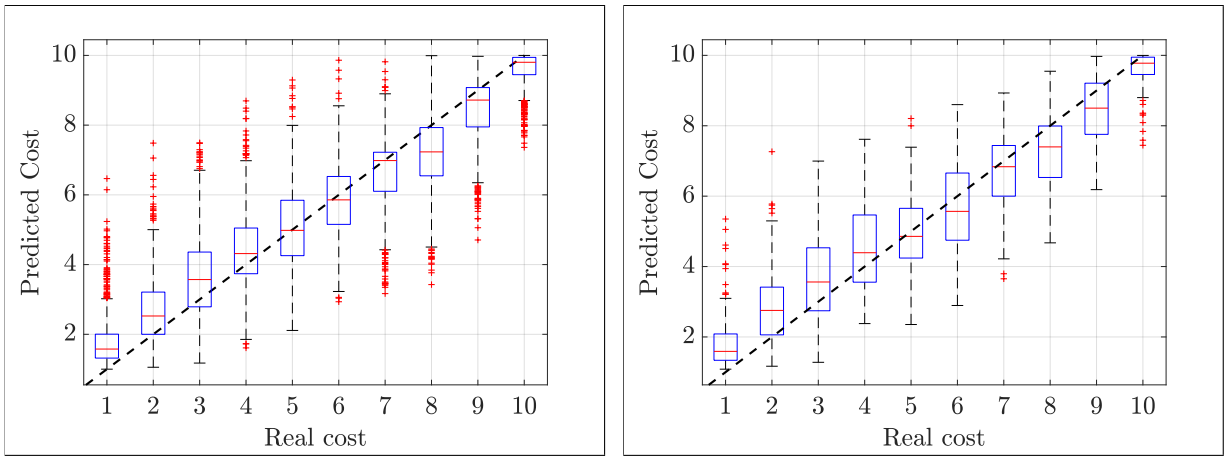


Figure 13: Plot showing the BIC value when fitting up to 50 Gaussians on the dataset. The minimum occurring at 24 indicates that this is the number of Gaussians that must be selected

To test the fitting, the data is separated into a train and validation set. The model is fitted on the train data. The trained model is used to predict the cost of the validation data using GMR. Fig.14b shows the result of this prediction. The predicted cost are centered around the correct values. However the mean error between the true cost and the predicted cost is of 1.19. This error is quite significant considering the scale of the data. Fig.14a represents the prediction of the cost for the training data. This plot shows that the distribution of the predictions is similar to the predictions of validation data. This indicates that over-fitting is limited and that the data is simply too complex to model with a smaller error.



(a) (b)

Figure 14: The boxplots show the result of using the fitted GMM model to predict the cost of a joint configuration. The black line represent the correct prediction, i.e. the predicted and true cost are the same.**(a)** shows the result of GMR on the training data and **(b)** on the validation data. The red dots represent the outliers of the data.

4.1.2 Minimizing the cost

This model is then used to optimize the cost by using gradient descent. An initial joint configuration is chosen and the gradient is then followed to minimize the cost. Fig.15 shows the distribution of ending costs for each starting cost. Every joint configuration with a same cost is taken and optimized, the final distribution of the costs is observed and plotted. One can notice that on average most of the points finish with a cost located between 3 and 4 even for configurations starting at a lower cost. This result is not satisfying as the goal is to reach a configuration of cost 1 for any starting position. The reasons for this behavior is not clear. Also SVM-SVR described in the following section obtains more satisfactory results and is therefore focused on.

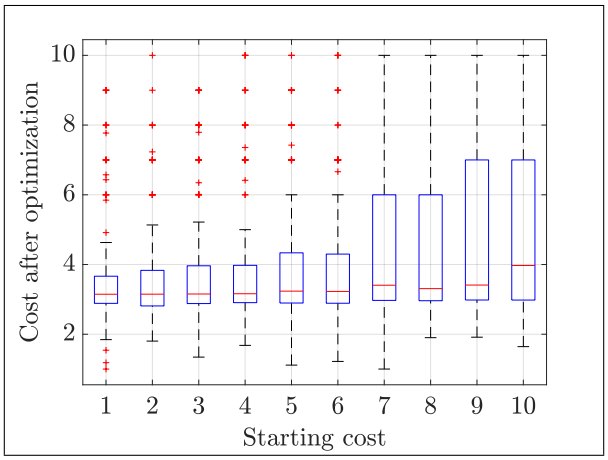


Figure 15: Evolution of the cost after optimization using GMMs. For each starting cost value the corresponding boxplots represents the distribution of the cost after optimization. For example, the joint configuration with a starting cost of 1 finish on average with a cost of 3. The red dots represent the outliers of the data.

4.2 Using SVM-SVR

This work is done using the code found at [20].

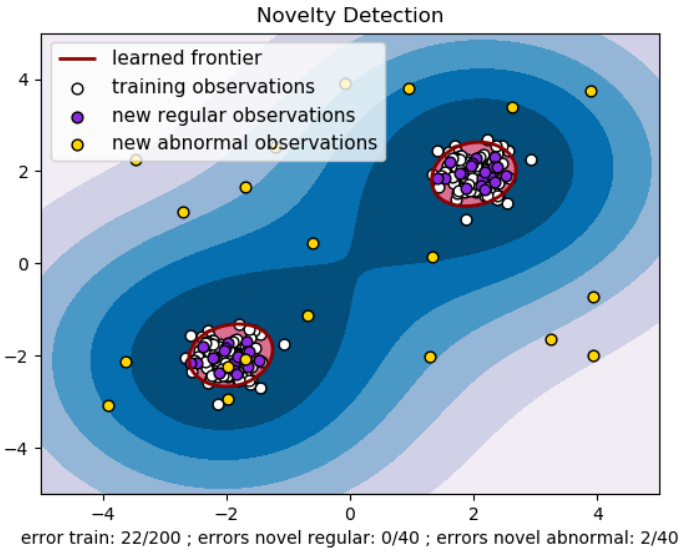
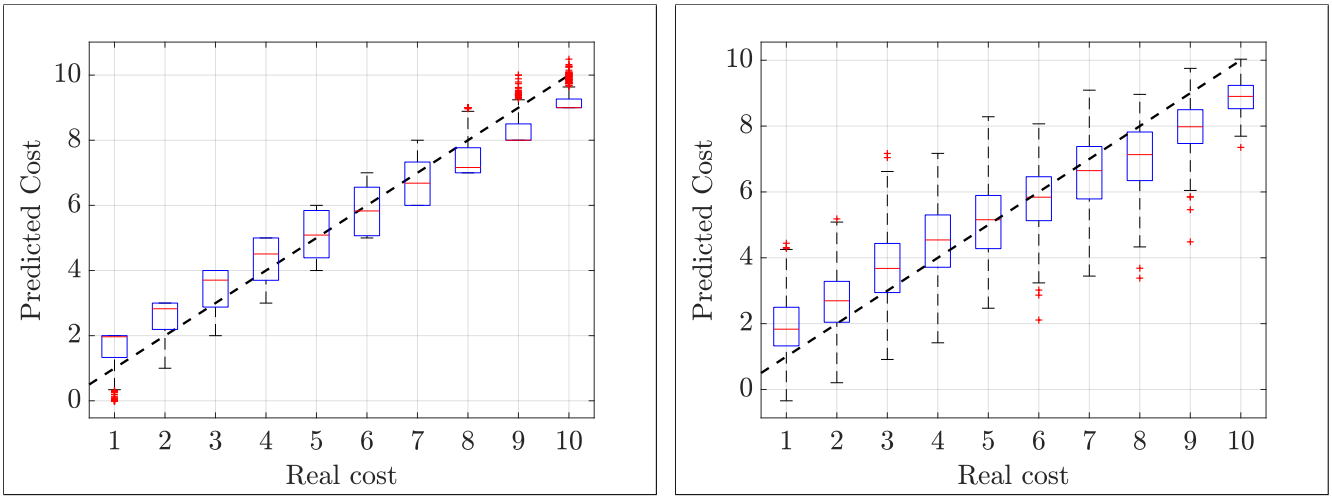


Figure 16: Example a one class SVM using a non-linear kernel [21].

4.2.1 Evaluating the cost of an unseen posture

The first SVR model used is a simple 10 class model representing the 10 different possible costs. The model is first fitted to the training data and is then used to predict the cost of the validation data. Fig.17a and Fig.17b show the result of predicting the cost of the training data and the validation data respectively. As before, the prediction of the validation data follows a linear trend. However, the mean error is once again relatively large with a value of 1.24. Also the training data is fitted more accurately than the validation data which indicates that the model is over-fitting and further tuning should be considered. However when using this model to minimize the cost, (see next section) the performance are satisfactory and therefore the model is not changed at the time of writing.



(a) (b)

Figure 17: The boxplots show the result of using the fitted SVR model to predict the cost of a joint configuration. The black line represent the correct prediction, i.e. the predicted and true cost are equal.**(a)** shows the result of SVR on the training data and **(b)** on the validation data. The red dots represent the outliers of the data.

4.2.2 Minimizing the cost

As before, the model is used to perform optimization of the cost from an initial joint configuration. This is done by computing the gradient at every step and moving in the direction of this gradient. Fig.18 shows an example of such an optimization. The cost is successfully minimized. However, the minimization goes too far and the joints diverge to unfeasible values. The cost goes below the minimum possible value of 1. To prevent this behavior, the optimization is simply stopped when the measured cost is below 1.

Fig. 19 shows the evolution of the cost after optimization for every starting configurations. The cost after optimization is determined by finding the point in the dataset that is closest to the finishing configuration. This is to ensure that the final configuration is feasible. One can observe that for the lower cost configuration (below 5) optimization has no effect on the cost. However, for configurations of higher initial cost, the algorithm is able to change the configuration such that the cost is reduced with the most significant effect being on starting configurations with costs large than 6.

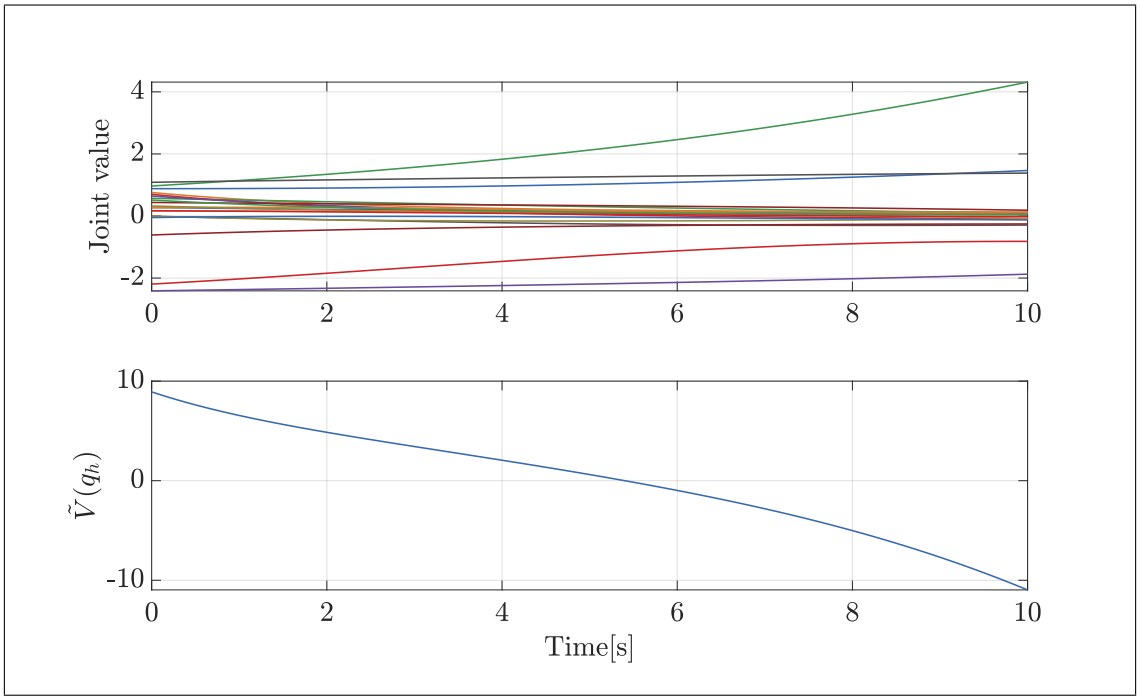


Figure 18: Example of minimization of the cost starting from a joint configuration with cost 8 when using a SVR model with 10 classes. One can observe that both the change in the cost and in the joints are smooth. However, the joints diverge and this results in a cost going below the smallest possible value of 1.

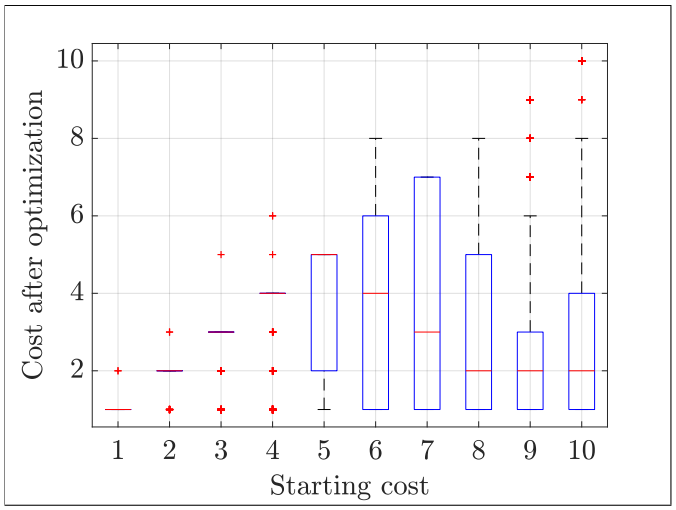


Figure 19: Evolution of the cost after optimization using SVR with 10 classes and stopping when the cost goes below 1. For each starting cost value the corresponding boxplots represent the distribution of the costs after optimization. The red dots represent the outliers of the data.

4.2.3 Minimizing the cost only considering only the end-effector

In this section, it is assumed that when the human is performing the task, he will adopt a configuration that is the most intuitive and the most comfortable given the context. For example when screwing on a object, he will not lift one leg off the ground or twist his spine completely. Therefore, only the end effector of the human is considered by assuming that all other joints are maintained in optimal configuration. In this case, the end effector is represented by the right-forearm as it is the one performing the task.

The dataset is modified so that only the x, y and z coordinates of the wrist and the elbow are kept. This allows to know the position of the hand as well as the orientation of the forearm. This new dataset is generated from the one described at the beginning of the chapter. However, it is not possible to simply remove the other joints and keep the associated costs unchanged. Indeed part of the cost is due to the other joints and when removing them the cost must be reduced. To achieve this, the data is clustered according to the 6 parameters that are kept. Then, for each cluster, the points that have a cost equal to the minimum of that cluster are kept and the others are discarded. This elitist choice ensures that the cost associated to a data point is due to the forearm. This leaves the dataset with a total of 2'281 points.

Using multi-class SVR

This modified dataset is then fitted with the a 10 class SVR. This model is then used to minimize the cost function from a starting forearm position described by the x, y and z coordinates of both the elbow and the wrist. Fig.20 shows an example of a minimization. As before, the estimated cost diverge to negative values due to the arm adopting configuration that are not feasible. In this case for example, the arm becomes longer as the cost goes down. To prevent this the length of the arm is maintained throughout the optimization. Fig.21 shows the result of an optimization when the length of the arm is maintained. The cost still decreases past 1 because the arm adopts a configuration with a low predicted cost that is not present in the dataset. For example the arm would go completely behind the back, a position infeasible by a human.

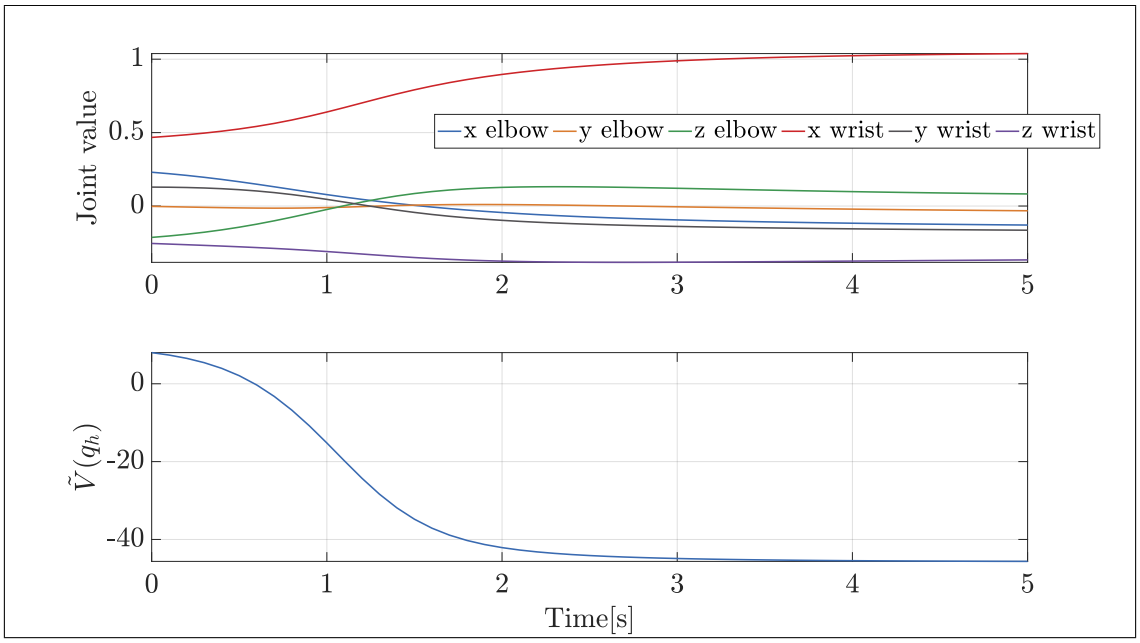


Figure 20: Example of minimization of the cost starting from a given forearm position with cost 8 when using a SVR with 10 classes. One can observe that both the change in the cost and in the joints are smooth. However, the cost decreases past the minimal value of 1. This due to the coordinates diverging to non-feasible configuration (elongation of the arm).

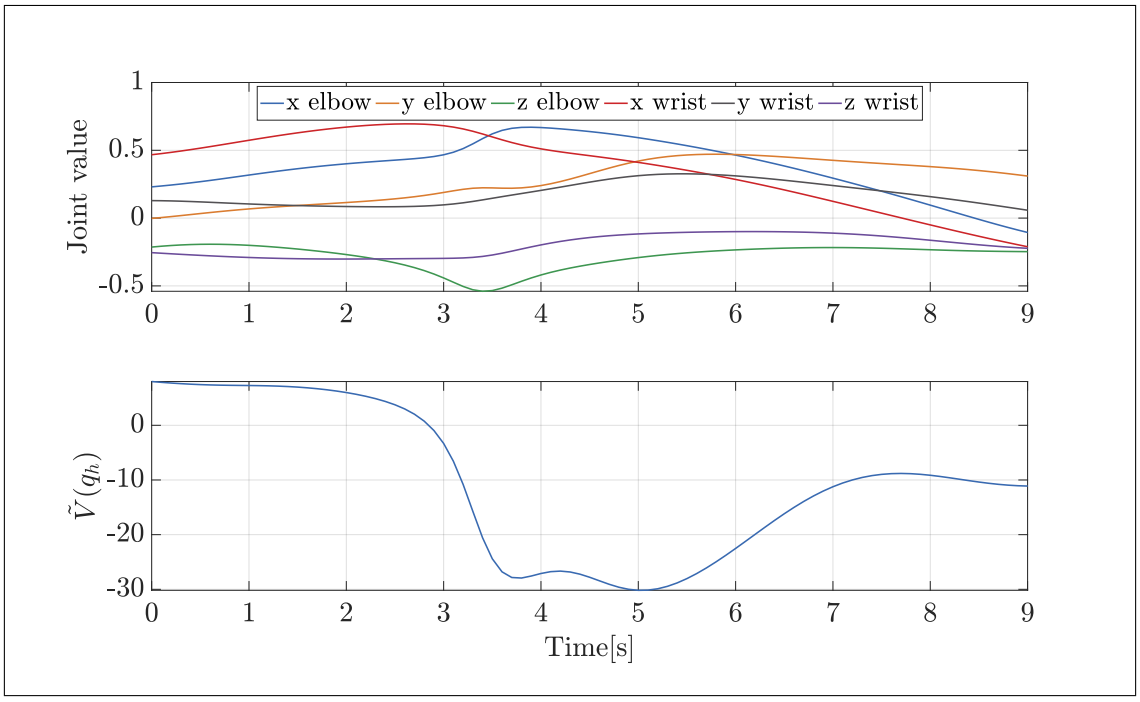


Figure 21: Example of minimization of the cost starting from a given forearm position with cost 8 when using a SVR with 10 classes. During the optimization, the norm of the arm is maintained. Nevertheless, the cost decreases past the minimal value of 1 once more. The forearm is now maintaining its length but it is going into unfeasible configuration (such as the arm completely behind the back) that have a predicted lower cost.

Using multi-class SVR combined with one class SVM

The problem that appears in the previous examples is that the joints go to infeasible configuration. These configurations have a lower predicted cost due to the structure of the SVR. This region of the space has no data points and therefore SVR extrapolates from what is sees resulting in false prediction. To counter this problem, a second model is created. This is a one class SVM (Fig.16) surrounding all the points in the dataset. This creates a border around the feasible points. The optimization is then done by adding the gradients of the two models. The first one to minimize the cost function and the second one to maintain the forearm in a feasible configuration.

Fig.22 shows the results of an optimization using the two models. One can observe that the cost no longer goes below 1 as the arm is maintained in the feasible set.

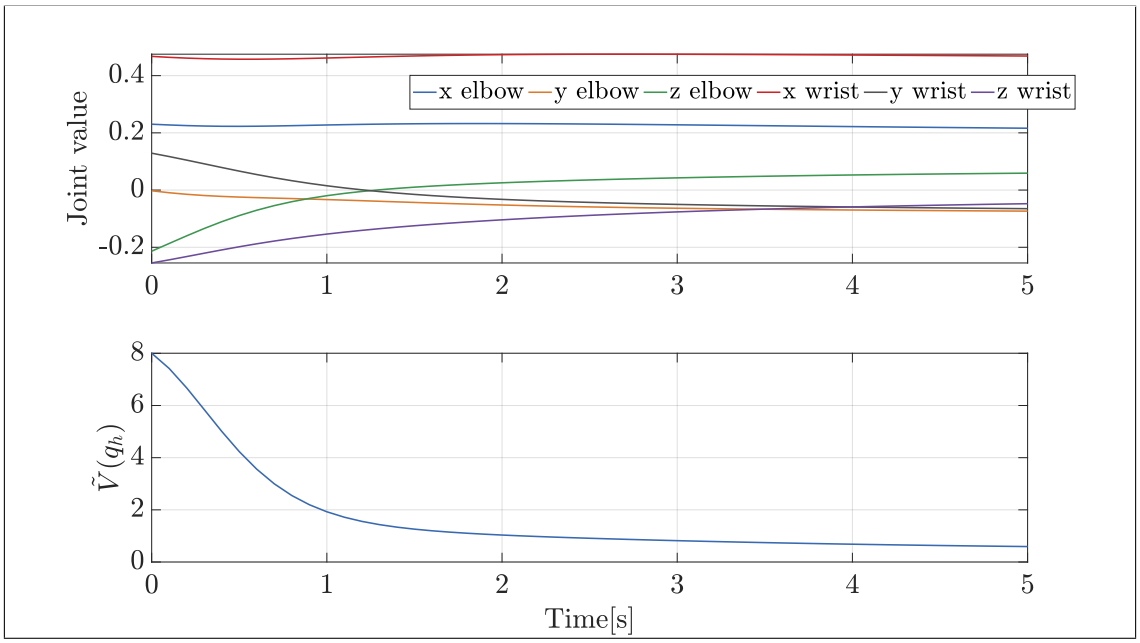


Figure 22: Example of minimization of the cost starting from a given forearm position with cost 8 when using a SVR model with 10 classes combined to a one class SVM model to maintain the configuration in a feasible set. During the optimization, the norm of the arm is maintained. The cost is successfully optimized to 1 without diverging.

Fig.23 shows the evolutions of the costs after optimization for the different starting configurations. The results have been improve compared to the previous method. Indeed, The cost is now even further reduced by the optimization. However, the optimizations that start at a configuration with cost 1 or 2 sometimes optimize to configurations with higher costs. This could be due to the fact that the SVM is an approximation of the actual cost and it brings the configuration to a location where the cost is actually higher than it believes.

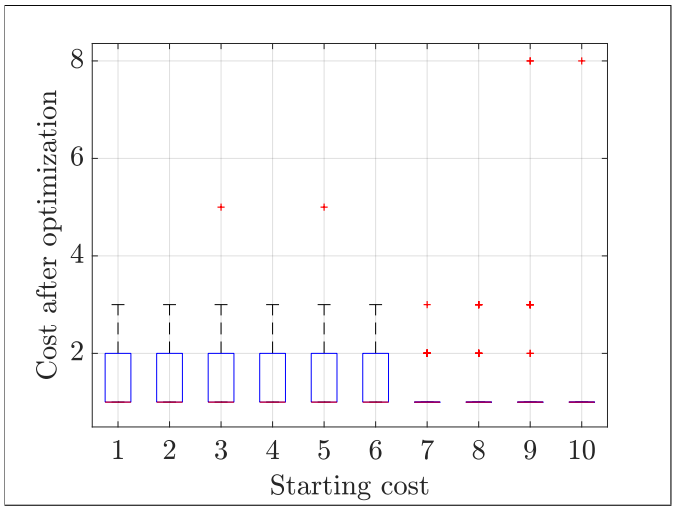


Figure 23: Evolution of the costs after optimization using 2 models. One SVR with 10 classes, one per possible cost. And one one class SVM representing the feasible set. The gradient of both models is added to minimize the cost. For each starting cost value the corresponding boxplots represent the distribution of the cost after optimization. The red dots represent the outliers of the data.

Using multiple one class SVM

Following on the idea of one class SVM, another method is explored. Here, 10 one class SVMs are used, one per cost. During minimization the goal is to go towards the class with a cost lower by one than the class the configuration is currently in. For example if the detected cost is 6 then the gradient to follow is the one of the SVM associated to a cost of 5. Fig.24 shows an example of such a minimization. The cost decreases incrementally. The gradient is no longer smooth due to sudden change in the cluster of attraction when reaching a new class.

Fig.25 shows the improvement in the cost after optimization for the different starting configurations. The plot shows that nearly every starting configuration leads to an optimal forearm posture after optimization. This method is therefore the method with the best results. However, this method assumes that there is a discrete cost available. This is not always the case and the previous methods remains a more general approach to the problem.

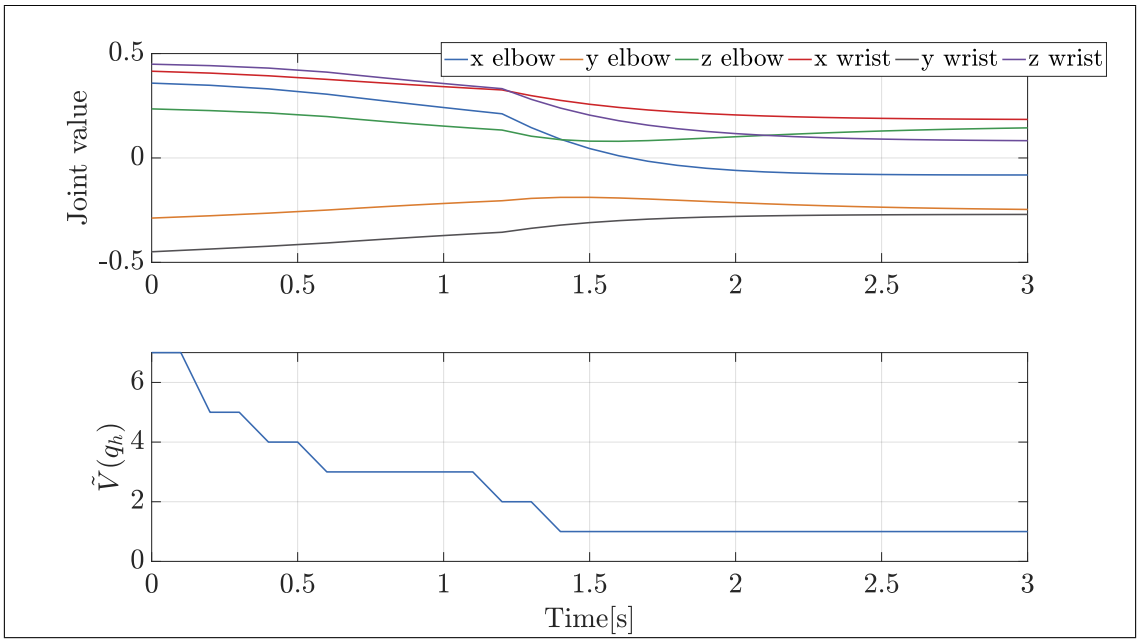


Figure 24: Example of minimization of the cost starting from a given forearm position with cost 7 when using 10 one class SVM model, one associated to each cost. The minimization is done in successive steps, when reaching a cost i, the gradient is changed to follow the gradient associated to the one class SVM of cost $i-1$. The cost is successfully optimized to 1. The gradient is no longer smooth.

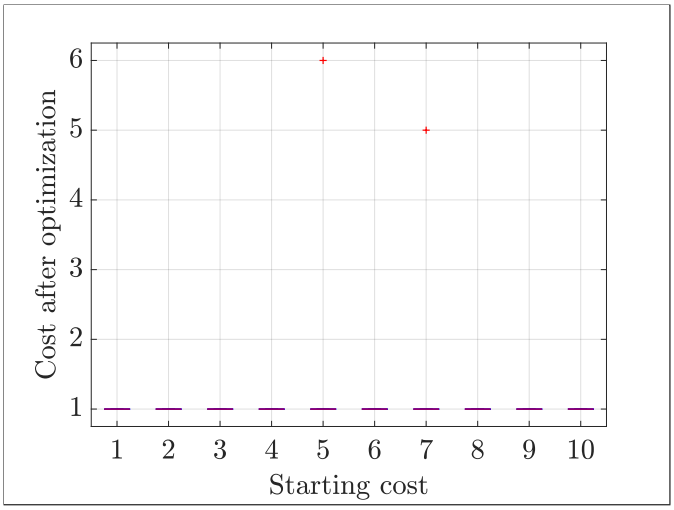


Figure 25: Evolution of the cost after optimization using 10 one class SVM models, one per cost. For each starting cost value the corresponding boxplots represent the distribution of the cost after optimization. This methods is efficient as nearly all starting configurations lead to a perfect score after optimization. The red dots represent the outliers of the data.

Chapter 5

Conclusion and future work

Conclusion

In this project, the possibility of improving the human posture through human-robot interaction was explored. The developed method focuses on associating a cost to human posture. The goal is then to minimize this cost by changing the joint configuration with gradient based methods. The cost of a given joint configuration is obtained by using the REBA method. This methods is a simple pen and paper method that associate a discrete cost to a fix posture using a set of lookup tables. This cost his however not derivable and therefore not suited for gradient base methods for minimization. To overcome this challenge the use of regression is explored. Two methods are considered, Gaussian Mixture Regression (GMR) based on Gaussian Mixture Models (GMM) and Support Vector Regression (SVR) based on Support Vector Machine (SVM).

Both methods are tested on a dataset of randomly generated poses. Test are done in simulation to see if the methods can predict the cost of an unseen configurations as well as minimizing the cost of the posture by moving the joints. Both techniques provide the same accuracy in terms of predicting the cost of new data points. However, when trying to minimize the functions, GMM is unable to provide satisfactory results when minimizing. On the other hand, SVM is able minimize the cost efficiently. 3 different approaches are tested with SVM and SVR:

- Using a single SVR with 10 classes, one per possible cost. This method is able to minimize the cost but the joints diverge to impossible configuration with negative cost. However, stopping the optimization when the cost gets to 1 proved to be sufficient to prevent diverging.
- Using 2 models, one to describe the data and one to describe the feasibility set. The first one is a SVR with 10 classes, one per possible cost. The second one is a one class SVM that takes all the configurations as input and classifies if a point belongs to the feasible set or not. On top of this constraints are added to maintain the structure of the joints. This second methods improves on the first one as it does not diverge anymore and is able to reduce the cost even further.
- Using 10 single class SVM classifiers, one per associated cost. The function is then minimized by successively following the gradients of SVM of lower cost. This method proves to be the most successful. However, because the cost must be expressed as a discrete variable, it lacks the ability to generalize to arbitrary cost.

Future work

The next step of this project should focus on applying what was shown in simulation to an actual robotic system and overcoming the challenges that will appear at that step. Also, more thought should be given to the GMM approach to see if it can work as well as the SVM approach or understand thoroughly why it is not suitable.

Antoine Laurens

Date and Signature

Bibliography

[1] Parisa Nejati et al. "The relationship of forward head posture and rounded shoulders with neck pain in Iranian office workers". In: *Medical journal of the Islamic Republic of Iran* 28 (2014), p. 26.

[2] Laura Punnett and David H Wegman. "Work-related musculoskeletal disorders: the epidemiologic evidence and the debate". In: *Journal of electromyography and kinesiology : official journal of the International Society of Electrophysiological Kinesiology* 14 (2004), pp. 13–23.

[3] Andrew Ng, Melanie Hayes, and Anu Polster. "Musculoskeletal Disorders and Working Posture among Dental and Oral Health Students". In: *Healthcare* 4 (2016), p. 13.

[4] Sue Hignett and L Mcatamney. "Rapid entire body assessment (REBA)." In: *Applied ergonomics* 31 2 (2000), pp. 201–5.

[5] Eya Barkallah et al. "Wearable Devices for Classification of Inadequate Posture at Work Using Neural Networks". In: *Sensors* 17 (2017), p. 2003.

[6] Luka Peternel et al. "Adaptive Control of Exoskeleton Robots for Periodic Assistive Behaviours Based on EMG Feedback Minimisation". In: *PLOS ONE* 11.2 (Feb. 2016), pp. 1–26. DOI: 10.1371/journal.pone.0148942. URL: https://doi.org/10.1371/journal.pone.0148942.

[7] B. Busch et al. "Postural optimization for an ergonomic human-robot interaction". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 2778–2785. DOI: 10.1109/IROS.2017.8206107.

[8] *worker*. http://www.elcosh.org/record/images/1128.jpg. Accessed: 2019-05-28.

[9] *assisting-robot*. https://thumbor.forbes.com/thumbor/960x0/https%3A%2F%2Fspecials-images.forbesimg.com%2Fdam%2Fimageserve%2F1097911782%2F960x0.jpg%3Ffit%3Dscale. Accessed: 2019-05-28.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B* 39.1 (1977), pp. 1–38.

[11] Zoubin Ghahramani and Michael I. Jordan. "Supervised learning from incomplete data via an EM approach". In: *Advances in Neural Information Processing Systems 6*. Ed. by J. D. Cowan, G. Tesauro, and J. Alspector. Morgan-Kaufmann, 1994, pp. 120–127. URL: http://papers.nips.cc/paper/767-supervised-learning-from-incomplete-data-via-an-em-approach.pdf.

[12] Sylvain Calinon. "Nonlinear Regression". In: Robot Learning & Interaction Group Idiap Research Institute, 2015.

[13] *REBA Table*. https://www.physio-pedia.com/images/a/a6/REBA.png. Accessed: 2019-06-01.

[14] *Kinect wikipedia*. https://en.wikipedia.org/wiki/Kinect.

[15] *Nuitrack*. https://nuitrack.com/.

[16] P. Song, Y. Yu, and X. Zhang. "Impedance Control of Robots: An Overview". In: *2017 2nd International Conference on Cybernetics, Robotics and Control (CRC)*. 2017, pp. 51–55. DOI: 10.1109/CRC.2017.20.

[17] *Human-movit-config repository*. https://github.com/baxter-flowers/human_moveit_config.

[18] *Mocap-optitrack*. https://github.com/epfl-lasa/mocap_optitrack.

[19] *Wikipedia BIC*. https://en.wikipedia.org/wiki/Bayesian_information_criterion.

[20] *github SVM repository*. https://github.com/nbfigueroa/SVMGrad.

[21] *One Class SVM Example*. https://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html.