Universiteit Leiden
The Netherlands

Opleiding Informatica

Robust self-balancing

robot mimicking

Robbin Borst

Supervisors:
E.M. Bakker
M.S.K. Lew

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

18/08/2017

Abstract

In this paper an imitating system is proposed that enables humanoid robots to imitate full-body human motions in real-time while keeping balance. The Windows Kinect 2.0 device has been used to acquire the coordinates of the joints of the human skeleton in real-time. Through a process of Inverse Kinematics the joint angles of the humanoid robot corresponding to the imitated human posture are calculated. This is done by minimizing an evaluation function that calculates the distance between the human posture and a robot posture calculated with Forward Kinematics. The process is similar to the process used in Ou et al. [8]. The resulting joint angles are called the target angles. The robot iteratively rotates its joints towards the latest obtained target angles. In each iteration balance maintenance is applied to ensure the stability of the robot during the imitation process using the *qpOases* [2] library in which all joints of the robot are considered. qpOases also ensures the supporting feet stay fixed to the ground. This balance maintenance system differs from the balance maintenance systems used in previous works. The process of calculating the target angles and the process of moving the robot towards the target angles run in parallel on separate threads of the processor. As in previous works, three different support modes are considered: Left-foot support mode, right-foot support mode and double-foot support mode. The experiments show the imitation system is capable of imitating complex human motions while keeping the robot in balance.

Contents

Chapter 1

Introduction

Robot imitation has multiple useful purposes in practise. Human-like humanoid movement is normally hard to achieve. Imitation can be used to learn a humanoid to move like a human. Furthermore, imitation can be used for entertainment purposes. Humanoid imitation can also be used for tele-operation between a human and a robot. An example of tele-operation through imitation is the robot in the movie *Avatar*. This robot performs human-like military movement, while the commander is located inside the robot. This robot is being developed in reality in South-Korea by the company *Hankook Mirae Technology* and is shown in Fig. 1.1.

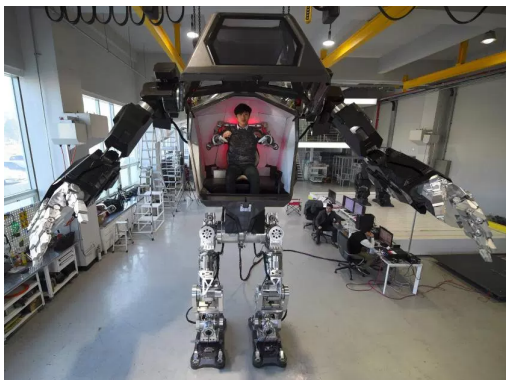

Figure 1.1: Avatar-like robot developed in South-Korea by Hankook Mirae Technology
Retrieved August 3, 2017 from
`http://leagueoftechnicalvoters.org/2017/01/01/giant-avatar-style-robot-takes-first-steps-in-south-korea/`.

There are multiple difficulties that have to be considered while developing a humanoid imitation system. First of all the desired joint angles of the robot's limbs have to be calculated to imitate the current human posture in real-time. As the range of motion of humanoids differs from the range of motion of humans, certain human motions might not be reachable for the humanoid. In this case a posture close to the human posture should be calculated and adopted by the robot. Moreover balance has to be maintained by the robot during the imitation process while standing on one foot or both feet. The balance maintenance system of the robot differs from that of the human: The robot has no muscles to maintain balance and its weight is distributed in a different way. Because of this the robot motion cannot be mapped one-to-one with human motion. Finally self-collision avoidance of the robot should be considered to prevent the robot from damaging itself.

The organization of the paper is as follows: In chapter 2 the relevant work is described and compared to our work. In chapter 3 the imitation process is explained, including the calculation of the target angles of the imitated posture and the balance maintenance system. In chapter 4 the setup that has been used to develop the imitation system is described. Finally in chapter 5 the experiments are shown and in chapter 6 the conclusions are drawn.

Chapter 2

Related Work

In this chapter an overview of related work is given and its relevance is compared and discussed relative to our method. Koenemann et al. [4] presented a human imitation system for humanoids in which whole-body motions are imitated in real-time. An *Xsens MVN* motion capture system was used to capture the human motions. This motion capture system consists of inertial sensors that are attached to the human body. This capture system is very accurate, but also expensive. The positions of the endeffectors, i.e. the hands and the feet, and the position of the center of mass of the robot were considered in the imitation process. The joint angles of the imitated posture of the robot are calculated through a process of *Inverse Kinematics* (IK). Afterwards, these angles are adjusted to match the position of the center of mass of the human. The supporting state of the robot is determined by a finite state machine, depending on the position of the center of mass of the robot, the human posture and the current support state. The support state can be either *left foot*, *right foot* or *both feet*. In order to shift the center of mass to the desired position in the double-foot support state, one of the feet is repositioned. In the single-foot support phase the joint angles of the corresponding leg of the supporting foot are adjusted to achieve a stable posture. The desired angles are calculated through Inverse Kinematics. The velocity of the center of mass is constrained. Their system was the first to imitate human motion sequences on a NAO robot, while balancing on a single foot for a longer period of time. Also experiments with tele-operation have been successfully performed. Ou et al. [8] developed a real-time full-body human imitation system with the use of the Microsoft Kinect 1.0. The Kinect is an easy-to-use motion capture system, of which the retrieved motion data can be processed with the *Kinect for Windows SDK*. Due to its user-friendliness and low pricing, it is an attractive choice. The angles of the imitated posture are calculated with Inverse Kinematics by minimizing an evaluation function. The robot moves iteratively towards the imitated posture. For each step balance control and collision avoidance are applied. In order to maintain the balance of the robot, the *ankle strategy* has been used in the single-foot support phase and the *ankle-hip strategy* has been used in the double-foot support phase. These balancing methods are explained in [9]. The process of calculating the joint angles and the balance maintenance system run in parallel on separate threads of the processor. The results of the experiments show the system is capable of imitating various complex human motions, in double-foot support phase as well as in single-foot support phase. Lei et al. [7] also developed a real-time full-body human

imitation system using the Kinect 1.0. When the supporting state is changed, a transient pose is transferred to the robot. There are three different transient poses: Standing straight, leaning to the left and leaning to the right. While this results in safer support mode changes, it is at the expense of the similarity between the motions of the human and the robot. The target robot posture is first calculated by a quadratic optimization process through Inverse Kinematics, similar to the optimization process in [8]. Next, the distance between the center of mass and the center of the current support polygon is minimized by modifying the angle list corresponding to the target posture through Inverse Kinematics. The angle list corresponding to the resulting stable posture is then transferred to the robot. This differs from article [8] in which balance control is applied iteratively during the movement of the robot towards the target pose. A *latent structure model* is used to study shared information between human motion space and robot motion space. This information is used to create a similarity function. The experiments show multiple human postures in single-foot support mode as well as in double-foot support mode have been successfully imitated.

In our work the Kinect 2.0 is used to capture human motion, which can track 26 skeleton joints in contrast to the Kinect 1.0 which can track 20 skeleton joints. Furthermore the field of view of the Kinect 2.0 is greater than the field of view of the Kinect 1.0. The *XSens MVN motion capture system* used in Koenemann et al. [4] is more accurate than the Kinect 2.0, however this motion capture system is expensive and harder to use than the Kinect 2.0. A similar process to calculate the imitated posture of the humanoid is used in our work as in the work of Ou et al. [8]. The Inverse Kinematics process in which an evaluation function is minimized proved to be accurate and efficient. The implemented balance maintenance system differs from previous work. The *qpOases* [2] library has been used to adjust the calculated imitated posture such that after modification the center of mass is positioned in the support polygon and the supporting feet stay fixed to the ground. This library considers all the joints of the humanoid to provide stability, similar to Lei et al. [7]. The balance maintenance system consists of multiple steps from the current posture towards the next imitated posture, similar to Ou et al. [8]. In the other two articles the balance maintenance system consists of only a single adjustment by modifying the target posture to provide stability. As in the previous works, both the double-foot support state and single-foot support state are considered. Thus at least one foot is fixed to the ground at any point. As in Ou et al. [8] multi-threading has been used to calculate the imitated postures and to move the robot towards the latest obtained angles corresponding to the imitated posture. In the other two articles these processes run in sequence. Finally, in our work the free foot is kept parallel to the ground as long as it is feasible by the range of motion of the humanoid. This makes it easier to put the foot back on the ground when requested and prevents collision of the free foot with the ground.

Chapter 3

The imitation process

The imitation process can be illustrated as shown in Fig. 3.1. Two repeating sequences can be seen in the diagram. In the first sequence, the digital skeleton of the human posture is acquired and the target angles of the corresponding imitated posture of the robot are calculated. The calculated target angles are sent to the second sequence. In the second sequence small steps are made by the robot to move the robot closer to the desired target angles. Each step results in a closer posture of the robot relative to the human posture. For each step, balance control is applied to ensure the stability of the robot. The two sequences run in parallel on separate threads of the CPU. In the following chapters the imitation process will be explained in detail.

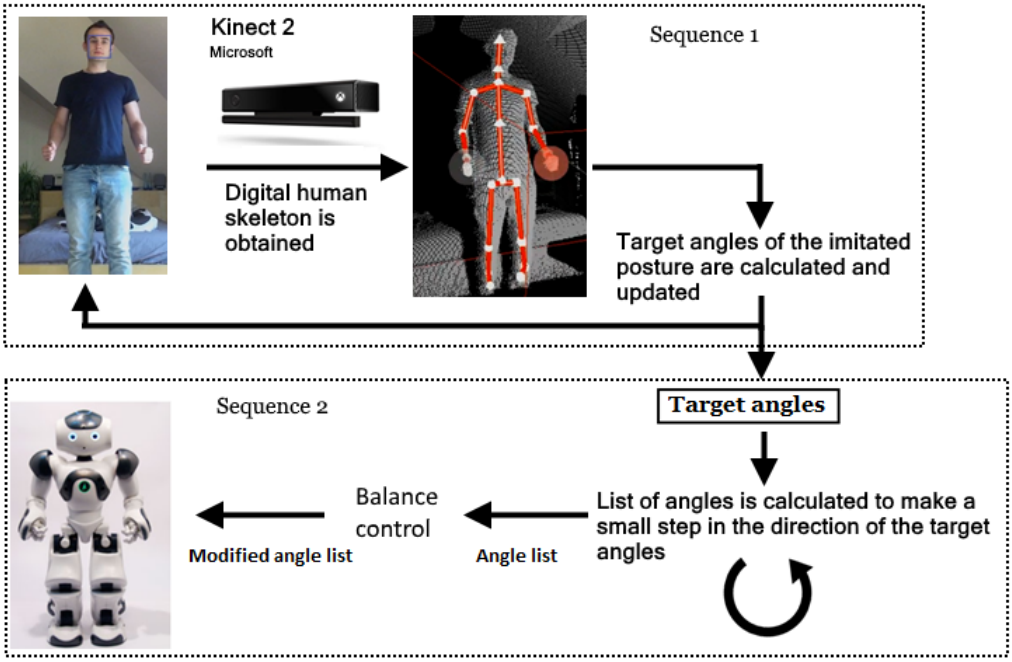

Figure 3.1: The imitation process.

3.1 Obtaining the digital human skeleton

The Microsoft Kinect 2.0 has been used to acquire digital human skeletons of the human posture in real-time at a frequency of 30 frames per second. An example of an acquired skeleton is shown in Fig. 3.2. The software development kit of the Kinect 2.0 is then used to extract the coordinates of the joints from the skeleton, visualized as white dots in Fig. 3.2. Thereafter, the extracted coordinates of the skeleton are transformed to the torso coordinate system. In the torso coordinate system, the torso is positioned upright and faces directly forwards. Finally, the resulting coordinates of the joints in the torso coordinate system are stored. If the human stands too close to the Kinect, the human skeleton might be deformed. Thus the imitation process is paused to prevent unwanted movements of the robot.

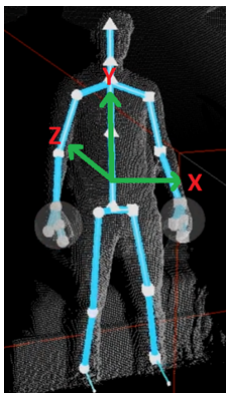

Figure 3.2: The digital human skeleton in the torso coordinate system.

3.2 Calculation of the target angles

The angles of the imitated posture of the robot are called the *target angles*. The target angles are calculated individually for each limb of the robot. The considered limbs are:

- The left arm.
- The right arm.
- The left leg.
- The right leg.

The head of the robot is not considered. The target angles are calculated through a process of Inverse Kinematics by minimizing an evaluation function. The evaluation function calculates the distance between the human posture and a robot posture. The coordinates of the posture of the robot in the torso coordinate system are calculated with Forward Kinematics from a list of joint angles and the lengths of the limbs. During the minimization process, a list of joint angles starting with predefined values is iteratively adjusted, such that the posture of the robot corresponding to the angles in this list will be closer to the human posture after each step.

When the minimization process ends, the resulting angle list contains the target joint angles. The previous target angles are replaced by the new target angles and the robot starts moving its joints towards the latest obtained target angles.

3.2.1 Evaluation function

The evaluation function calculates the error between the coordinates of a certain limb of the acquired human posture and those of the corresponding limb of the robot. The coordinates of the limb of the robot are calculated with the use of Forward Kinematics from the joint angles (Θ). Both coordinates of the human and the robot are located in the torso coordinate system. The coordinates of the joints of the human and the robot must be normalized, as the size of the robot differs from the size of the human. Normalization causes the Euclidean distance from the zero point to the concerning joint to be one. For the arms, the shoulder is the zero point and for the legs, the hip is the zero point. The proportions of the lengths of the body parts of the robot are similar to those of a human being and thus an accurate comparison between the postures can be made after normalization. The error is now calculated by the following formula:

$$\mathbf{E} = (\sum_{i=1}^{joints} dist({}^{T}H[i], {}^{T}R[i]))^2 \tag{3.1}$$

Where ${}^{T}H[i]$ is the normalized coordinate of joint i of the given limb of the human skeleton in the torso coordinate system and ${}^{T}R[i]$ the corresponding coordinate of the robot. $\mathbf{E}$ is the error, which is the summation of distances between all pairs of corresponding coordinates of the given limb, to the power of two. The pseudo-code of the evaluation function is shown below.

Algorithm 1 The evaluation function

```
procedure EVALUATE(Θ)
    ᵀR ← forwardKinematics(Θ)
    ᵀR ← normalize(ᵀR)
    ᵀH ← normalize(ᵀH)
    E ← 0
    for each joint ∈ Θ do
        E ← E + dist(ᵀH[joint], ᵀR[joint])
    end for
    return E²
end procedure
```

3.2.2 Minimizing the error

The minimizing problem can be defined as follows: find for each limb of the robot a list of joint angles for which the evaluation function returns a minimal error. In order to solve this problem, the Inverse Kinematics Levenberg-Marquardt (LM) algorithm is used, also known as damped least-squares (DLS) [1]. This algorithm was also used in article [8]. The algorithm starts with a list of joint angles of a specific limb with predefined

values and is iteratively adjusted with the aim to reduce the error returned by the evaluation function. In each step of the algorithm, the list of angles is adjusted by applying the following formula:

$$\Theta_{i+1} = \Theta_i + (\mathbf{J}^T \cdot \mathbf{J} + \lambda \cdot \mathbf{I})^{-1} \cdot \mathbf{J}^T \cdot -\mathbf{E} \tag{3.2}$$

Where Θ_i is the current list of joint angles and Θ_{i+1} will be the new list of joint angles. The angles are limited to stay within the range of motion of the corresponding joint angle values. $\mathbf{J}^T$ is the transpose of the *Jacobian* matrix. λ is the convergence variable. λ also prevents matrix singularities to occur, in which case it is not possible to calculate the matrix inverse [1]. $\mathbf{I}$ is the identity matrix of the result of $\mathbf{J}^T \cdot \mathbf{J}$. $\mathbf{E}$ is the error value of the concerning limb returned by the evaluation function. The Inverse Kinematics process ends when the error is below a certain fixed minimum or when a maximum number of iterations have been executed.

Obtaining the Jacobian matrix

The Jacobian matrix contains the first-order partial derivatives of a certain function. In our case, the first-order partial derivatives are approximated by calculating the local velocity of the error with respect to a change in joint angle value: $\frac{\Delta \mathbf{E}}{\Delta \alpha}$. The process of filling the Jacobian matrix is demonstrated by the following pseudo-code.

Algorithm 2 Calculating the Jacobian Matrix

```
procedure FILLJACOBIAN(Θ)
    E_curr ← evaluate(Θ)
    for each joint ∈ Θ do
        α ← Θ[joint]
        Θ[joint] ← Θ[joint] + β
        E_new ← evaluate(Θ)
        Θ[joint] ← α
        J[joint] ← E_new − E_curr
    end for
end procedure
```

Θ is the list of current joint angle values of the corresponding limb. β is a small value that indicates the change in joint angle value, which is set to 0.5 degrees.

The Inverse Kinematics process

The Inverse Kinematics error minimization process is demonstrated by the following pseudo-code.

Algorithm 3 The Inverse Kinematics process to calculate the target angles

procedure INVERSEKINEMATICS(Θ_{start}^{limb})
  $\lambda \leftarrow \lambda_{start}$
  $\mathbf{E_{curr}} \leftarrow evaluate(\Theta_{start}^{limb})$
  $\Theta_i \leftarrow \Theta_{start}^{limb}$
  $fillJacobian(\Theta_i)$
  for $i \leftarrow 0$ `to maximum iterations` **do**
    $\Theta_{i+1} \leftarrow \Theta_i + (\mathbf{J}^T \cdot \mathbf{J} + \lambda \cdot \mathbf{I})^{-1} \cdot \mathbf{J}^T \cdot -\mathbf{E_{curr}}$
    $\mathbf{E_{new}} \leftarrow evaluate(\Theta_{i+1})$
    if $\mathbf{E_{new}} < \mathbf{E_{curr}}$ **then**
      $\Theta_i \leftarrow \Theta_{i+1}$
      if $\mathbf{E_{new}} < \epsilon$ **then**
        `break`
      end if
      $\lambda \leftarrow \lambda / 2$
      $\mathbf{E_{curr}} \leftarrow \mathbf{E_{new}}$
      $fillJacobian(\Theta_i)$
    else
      $\lambda \leftarrow \lambda \cdot 2$
    end if
  end for
  $\Theta_{target}^{limb} \leftarrow \Theta_i$
end procedure

ϵ is a fixed value, in our case it is set to 10^{-5}. If the error gets below this value, the Inverse Kinematics process is stopped. In our work, λ starts with a value of 0.02. The maximum number of iterations is set to 100 to ensure low calculation time. Θ_{start}^{limb} is first set to the joint angles of the current limb of the robot posture. The LM algorithm only finds a local optimum, which is not necessarily the global optimum. If the retrieved error corresponding to the local optimum is greater than a certain threshold, the algorithm is restarted with a list of angles with different predefined values.

3.2.3 Summarizing the target angles calculation process

The process of acquiring the human skeleton and calculating the corresponding target angles is summarized by the following pseudo-code.

Algorithm 4 The process of acquiring the target angles

```
procedure calculateTargetAngles
    while true do
        humanFrame ← nextHumanFrame()
        if dist(humanFrame) < minAllowedDist then
            continue
        else
            ^T_H ← transformToTorsoCoordinateSystem(humanFrame)
            for i ← 0 to TOTALLIMBS do
                inverseKinematics(Θ^i_curr)
            end for
            updateTargetAngles(Θ_target)
        end if
    end while
end procedure
```

3.3 Moving the robot towards the target angles

The second part of the imitation process consists of moving the robot towards the target angles iteratively while keeping balance. In this process small steps are made from the current posture of the robot to the desired posture of the robot. For each small step the desired amounts with which the angles should be rotated are stored in an angle list. A maximum joint velocity is maintained to guarantee safe motion and stability. The proportions between angle velocities of the joints are maintained when a modification has to be made to achieve the maximum velocity. Hereafter, the list of angles is modified by the balance controller to ensure the resulting posture of the current step of the process is stable. The modification guarantees several conditions:

- The supporting feet stay fixed to the ground.
- The modification is minimal such that the similarity between the human posture and the robot posture is maintained.
- The calculated posture is within the range of motion of the robot.
- In the projection of the coordinate system onto the xz-plane, the center of mass is located inside the support polygon.

Finally, the resulting list is sent to the robot, which will rotate its joints accordingly. Three support modes are considered by the balance controller: The double-foot support mode, left-foot support mode and right-foot

support mode. The left-foot support mode and right-foot support mode comprise the single-foot support mode. The balancing process will be explained in the next chapter.

3.4 Balance control

3.4.1 The considered support modes

Single-foot support mode

In the single-foot support mode, the robot balances on a single leg. The projection of the center of mass of the robot on the ground is forced to stay within the support polygon of the supporting foot. An example of a balanced posture in single-foot support mode can be seen in Fig. 3.3. If the projection of the center of mass on the ground is located outside of the supporting foot, the robot will tilt. The supporting foot is constrained to stay fixed to the ground.

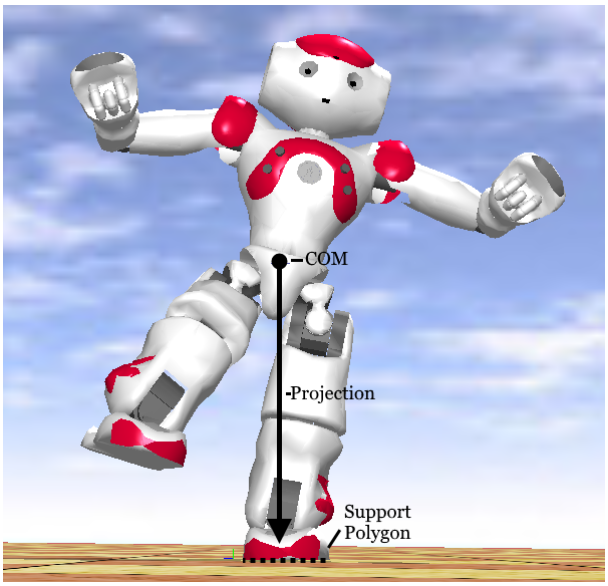

Figure 3.3: Balance maintenance in the single-foot support mode.

Double-foot support mode

In the double-foot support mode, both legs are used to maintain balance. The support polygon is the entire area between the two feet, as shown in Fig. 3.4. The projection of the center of mass on the ground has to stay within the support polygon at all time, or the robot will tilt. The robot is most stable when the center of mass projects on the straight line that connects the two ankles of the robot [8]. Both feet are constrained to stay fixed to the ground.

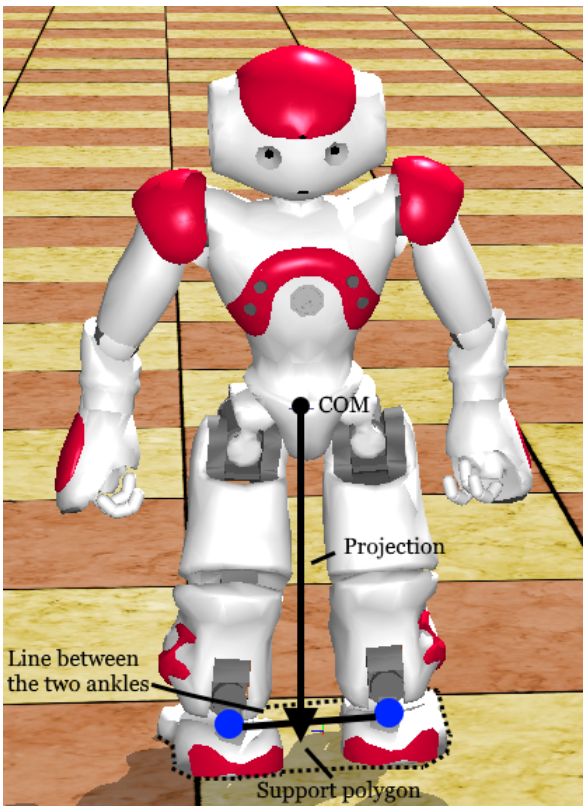

Figure 3.4: Balance maintenance in the double-foot support mode.

3.4.2 Modification of the angle list

The process of Inverse Kinematics is used to modify the angle list to maintain the balance of the robot. The Inverse Kinematics process can be written as a quadratic program, which is solved every 20 milliseconds using the open source library *qpOases* [2]. The classical form of a quadratic program is:

$$min \frac{1}{2}||Y - Y^{des}||_Q^2 \quad s.t. \begin{cases} AY + b = 0 \\ CY + d \geq 0 \end{cases} \tag{3.3}$$

Where Y^{des} is the desired solution. Y is the solution to be calculated such that the requirements described in chapter 3.3 are met. Y and Y^{des} contain the joint angles of the robot. Q is a given real symmetric matrix. A, b, C and d are the matrices and vectors that express the linear equality and inequality constraints respectively. In this case, the equality constraint is keeping the concerning feet fixed to the ground and the inequality constraints are keeping the projection of the COM on the ground inside the support polygon and keeping the joint values within the joint value limits. The coordinates of the center of mass can be calculated with Forward Kinematics from the joint angles, total mass of the robot, mass of the parts of the robot and lengths of the parts. The process of calculating the coordinates of the center of mass is explained in article [5]. Additionally, in the single-foot support mode, the foot of the free leg is kept parallel to the ground by adjusting the angles of the ankle, as long as it is feasible by the range of motion of the ankle. This makes it easier to safely put the free foot back on the ground when requested. The angles of the free foot relative to the ground can be calculated

from the joint angles in the kinematic chain from the support foot to the free foot. With this information, the needed modification of ankle joint angles in the current angle list can be calculated. The final modified angle list is sent to the robot, which rotates its joints accordingly.

3.4.3 Transitions between support modes

The next support mode of the robot is determined by the desired support mode, derived from the last obtained human skeleton, the current support mode of the robot and the positions of the feet of the robot. If the difference in height between the left foot and right foot of the human skeleton is greater than a certain threshold, in our case 15 centimetres, the desired support mode is set to be single-foot support mode. This will be left-foot support mode if the left foot is positioned lower than the right foot and right-foot support mode in the opposite case. Otherwise the desired support mode is set to be double-foot support mode. The diagram shown in Fig. 3.5 shows how the next support mode of the robot is determined. When a transition is made from double-foot support mode to one of the single-foot support modes, the COM of the robot is first shifted to the concerning support foot. This is done by applying the double-foot support mode balance control, with the COM constrained inside the support polygon of the concerning supporting foot. A transition from one of the single-foot support modes to the double-foot support mode can only be done when both feet of the robot are on the ground. There are a number of requirements that have to be satisfied before the free foot is put back on the ground:

- The free foot is four centimetres or less from the ground.
- The current support mode differs from the desired support mode.
- The polygons of the feet projected onto the xz-plane do not intersect.

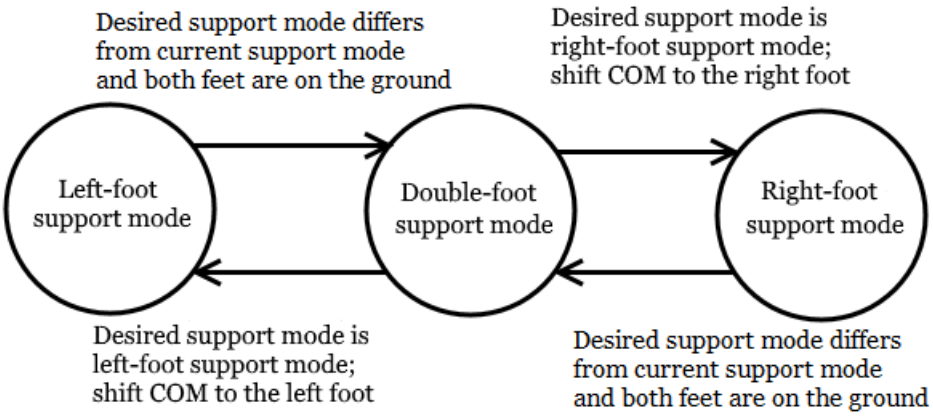

Figure 3.5: Determination of transitions between the support modes of the robot.

Chapter 4

The setup

4.1 The NAO robot

The humanoid robot that has been used to imitate the human is the NAO robot. The robot is described in the following chapters.

4.1.1 The robot in general

NAO is a robot made by the company Aldebaran Robotics. NAO has a total height of 58 centimetres, which is approximately one-third of the average height of an adult human being. The robot weighs 4.3 kilograms. NAO is an advanced robot with 25 degrees of freedom (DOF), as can be seen in Fig. 4.1. The *HipYawPitch* actuators are coupled. In other words, the NAO robot can perform 25 independent motions. This makes it possible for the robot to imitate complex human postures.

4.1.2 Sensors

NAO is equipped with many sensors. The location of the sensors are shown in Fig. 4.2. The robot contains two HD cameras, which are located in the forehead. The cameras can be used for object recognition. Four microphones are located in the head of the robot, which are used to capture sound in the environment. This makes it possible for the robot to recognize human voices and to be commanded by human spoken text. A loudspeaker is located in each ear, which enables the robot to speak. Two sonar sensors are located in the torso, with which the distance to obstacles in the environment can be measured. This can be used to prevent the robot from colliding against objects and possibly damaging itself. NAO also contains two infra-red emitters and receivers, located in the eyes. This makes it possible to use NAO as a remote control or receive orders from another remote control, which can be another NAO robot. An inertial board is located in the torso. This inertial board contains two gyrometers and an accelerometer. Furthermore, NAO contains nine tactile sensors.

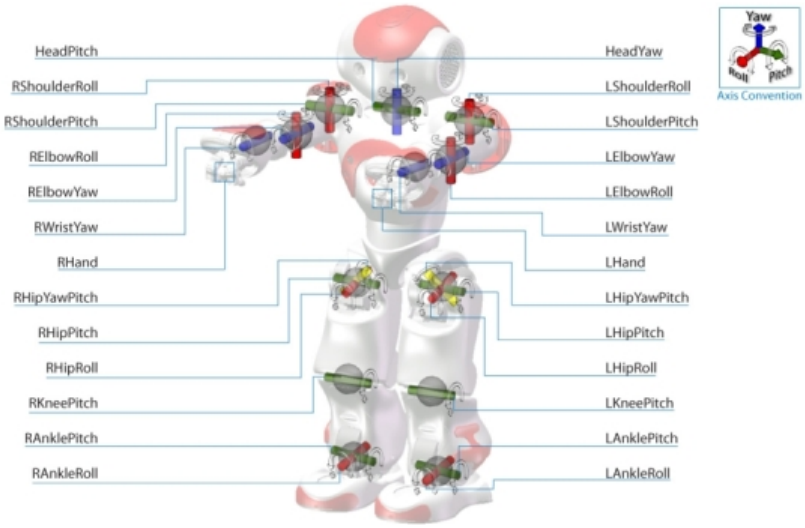

Figure 4.1: Actuators of NAO
Retrieved May 22, 2017 from `http://doc.aldebaran.com/2-1/family/nao_dcm/actuator_sensor_names.html`.

For example, the tactile sensors in its hands enable the robot to know when its hand is touched. Finally, four pressure sensors are located in each foot to measure the distribution of the weight on its feet.

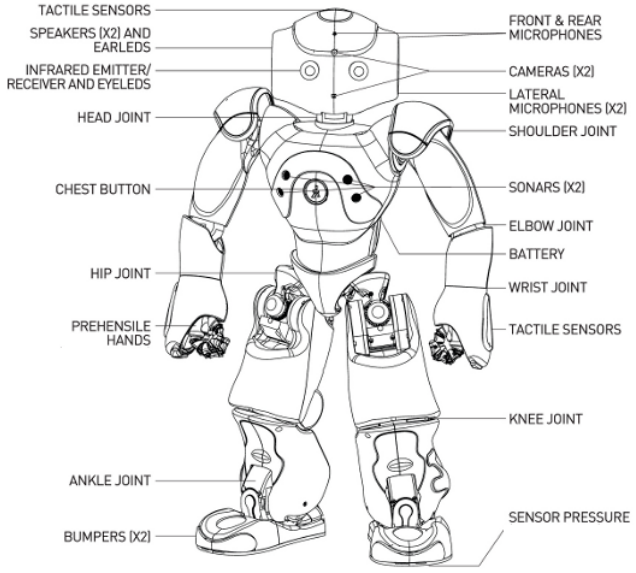

Figure 4.2: Sensors of NAO
Retrieved May 22, 2017 from `http://doc.aldebaran.com/1-14/family/nao_h25/index_h25.html`.

4.1.3 NAOqi Framework

NAO comes with an extensive framework that is used to program the robot. The framework is cross-platform and cross-language. It contains both a C++ API and a Python API. The API's contain functionality for motion, audio, vision, people perception, sensor communication, balance maintenance and more.

4.2 Programming setup

The programming language that has been used to write the imitation program is C++. The Windows SDK 2.0 has been used to acquire the human skeleton data from the Kinect data. The *Eigen* library [3] has been used for the matrix calculations. The *NAOKinematics* library [6], written by N. Kofinas, has been used to solve the Forward Kinematics of the NAO robot. As this library does not support Forward Kinematics of joints that are not located at the end points of the limbs, i.e. the knees and the elbows, modifications had to be made. The *Whole Body control* API of the NAOqi framework has been used for the balance maintenance of the robot, which uses the qpOases open source library described in chapter 3.4.2.

4.3 Simulators

The simulator *Choregraphe*, made by Aldebaran Robotics, has been used while developing the imitation system, before the balance maintenance system had been implemented. This simulator does not contain gravity. During the development of the balance maintenance system, a simulator that contains gravity was needed. The chosen simulator that meets this requirement is *Webots*, made by the company Cyberbotics. Webots is a professional simulator used in business as well as in research and education. The two simulators are shown in Fig. 4.3 below.

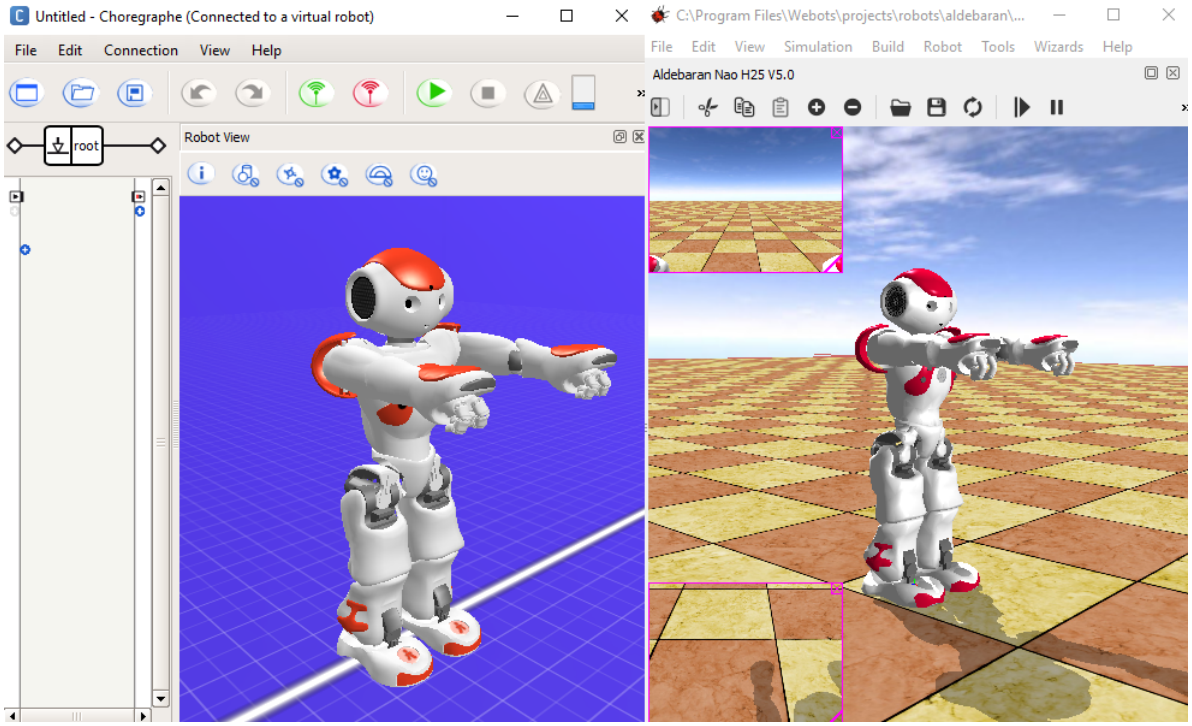

Figure 4.3: Simulators Choregraphe (left) and Webots (right).

Chapter 5

Experiments

5.1 Experiments on a virtual robot in simulator Webots

Similarity between human motion and robot motion

Four poses, shown in Fig. 5.1, have been chosen to measure the similarity between the human motion and robot motion. These postures cover the biggest part of the range of motion of the NAO robot in the double-foot support state. First the angle trajectories of the robot and the human are measured during the consecutive

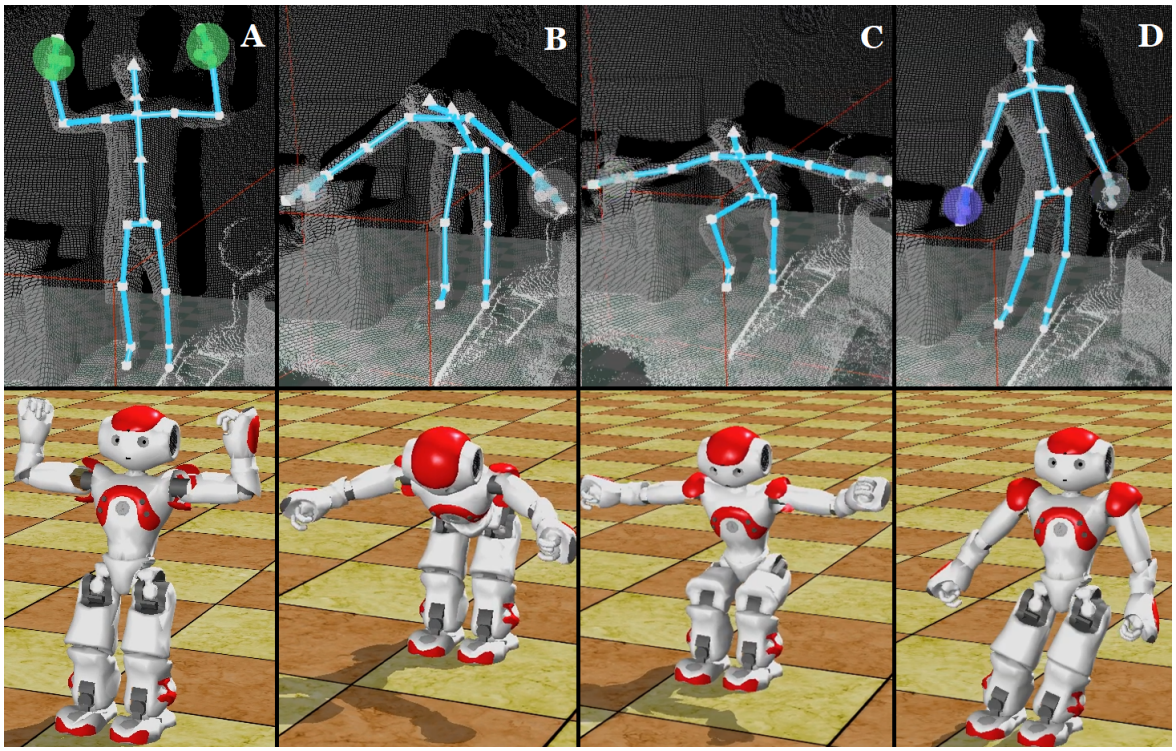

Figure 5.1: The four poses that have been selected to evaluate the similarity between the human motion and robot motion.

performance of poses A, B, C and D. The robot starts from the zero posture, in which all its joint angles are set

to zero. The results are shown in Fig. 5.2. Additionally the offsets of the arms, elbows, knees and ankles of the robot from the desired location are measured for the same motion sequence. The results are shown in Fig. 5.3. As can be seen in the graphs in Fig. 5.2, the angle trajectories of the limbs of the robot closely follow the angle

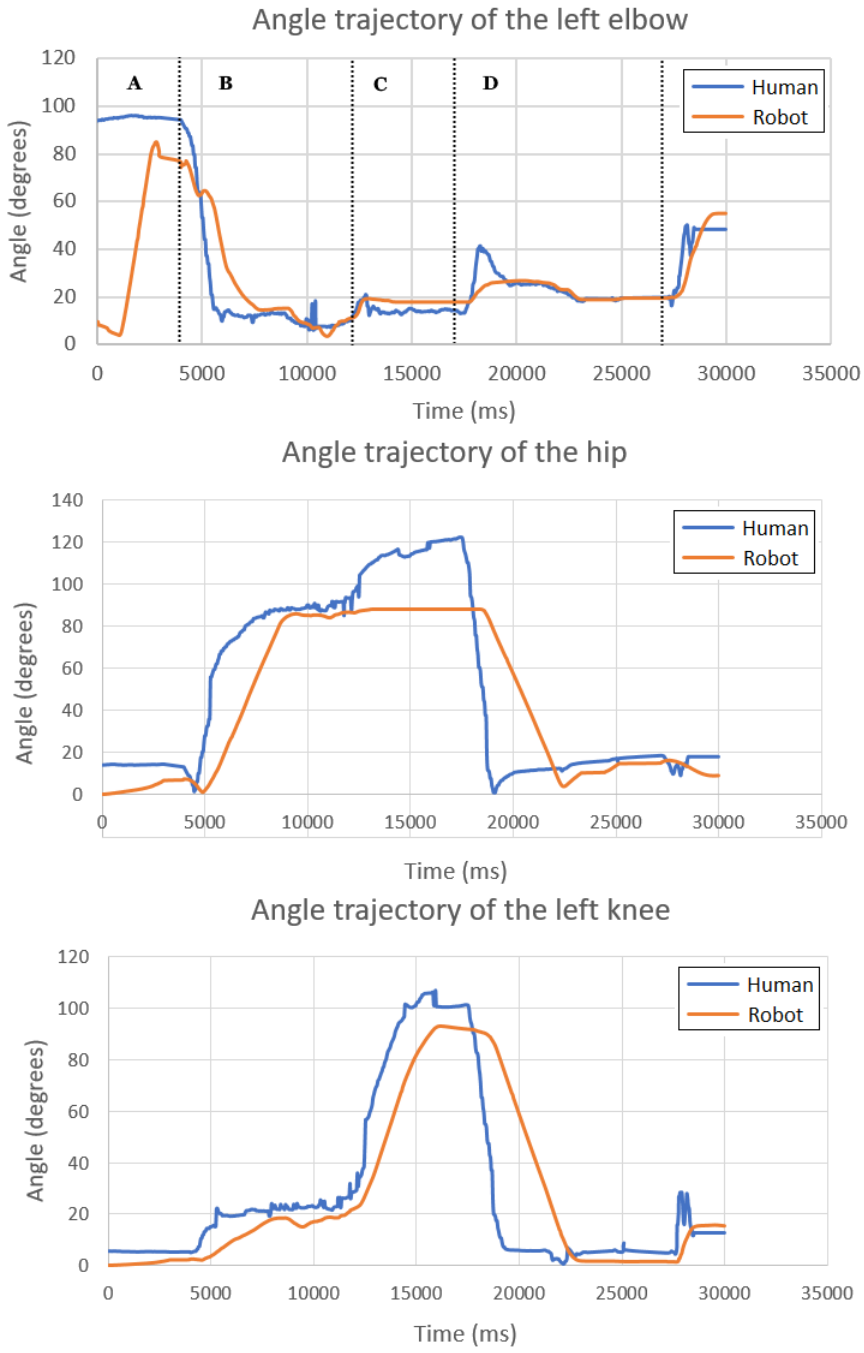

Figure 5.2: The angle trajectories of the left elbow, the hip and the left knee of the human and the robot during the consecutive performance of the four poses illustrated in Fig. 5.1 over time.

trajectories of the limbs of the human over time. The Levenberg-Marquardt algorithm effectively calculated the imitated postures of the humanoid by minimizing the evaluation function allowing the robot to imitate the human by rotating its joints. In the graph of the angle trajectory of the hip, one can observe the robot reaches its hip joint angle value limit while imitating posture C at circa 13000 ms and is thus not able to reach the hip joint angle value of the human. Also in the graph of the left knee, at circa 18000 ms one can observe the

robot reaches its maximum joint angle velocity and is not able to follow the human with the same velocity. The delay between the human angle and robot angle is caused by maximum joint angle velocities, balance control modifications and computation time. The graphs in 5.3 show the imitation system effectively reduces

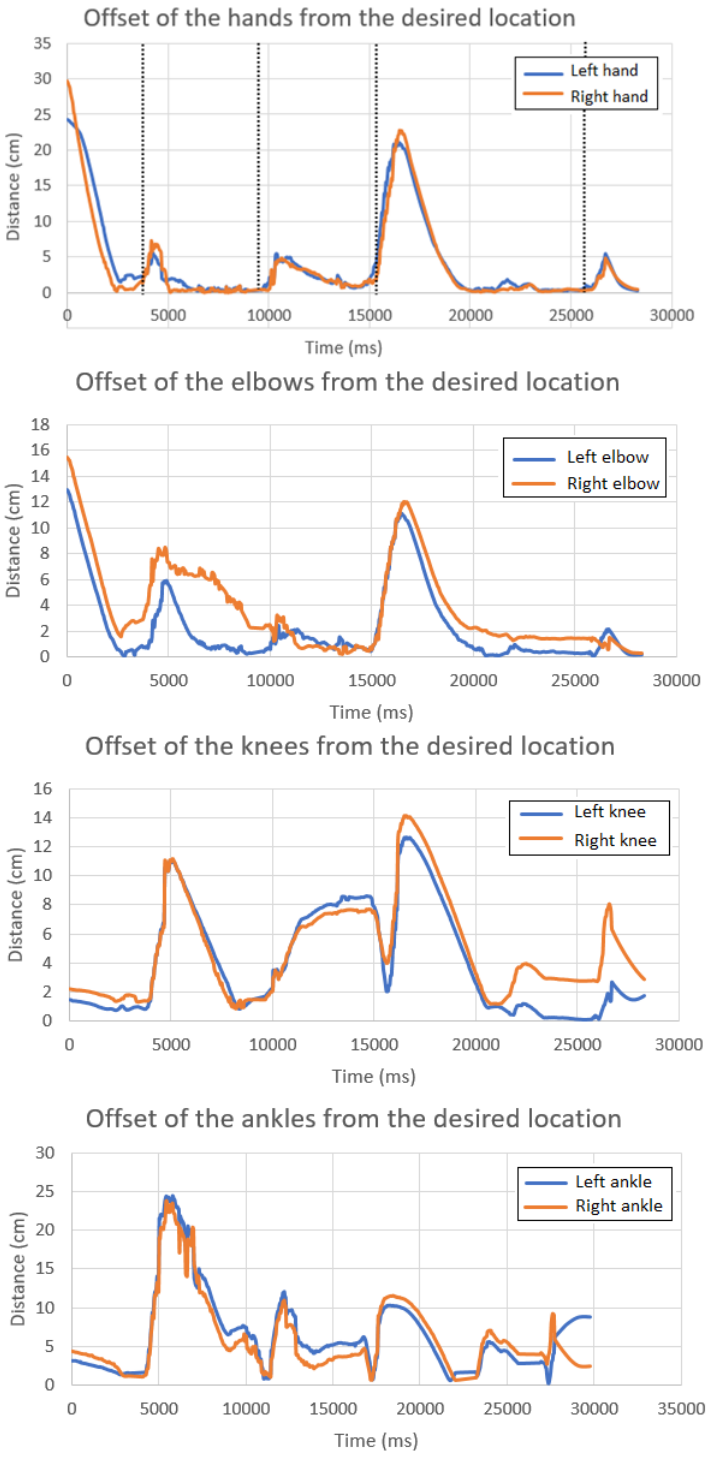

Figure 5.3: Offsets of the hands, elbows, knees and ankles of the robot in centimetres from the desired locations during the consecutive performance of the four poses illustrated in Fig. 5.1 over time.

the offsets of the joint locations from the desired locations during the imitation of postures A, B, C and D. The desired locations are the normalized coordinates of the human joints relative to the zero point of the limb.

When another posture is adopted by the human, the desired locations change from position, resulting in an increase of the offsets between the robot joints and the desired locations. At the end of each execution of the postures, shown by the dotted lines, the offsets have been effectively reduced.

Effect of the balance controller

To measure the effect of the balance controller on the stability of the robot, five poses illustrated in Fig. 5.4 have been selected. Balance is important during the consecutive performance of these five poses to prevent the robot from falling. During the consecutive performance of the poses, the distance between the left ankle

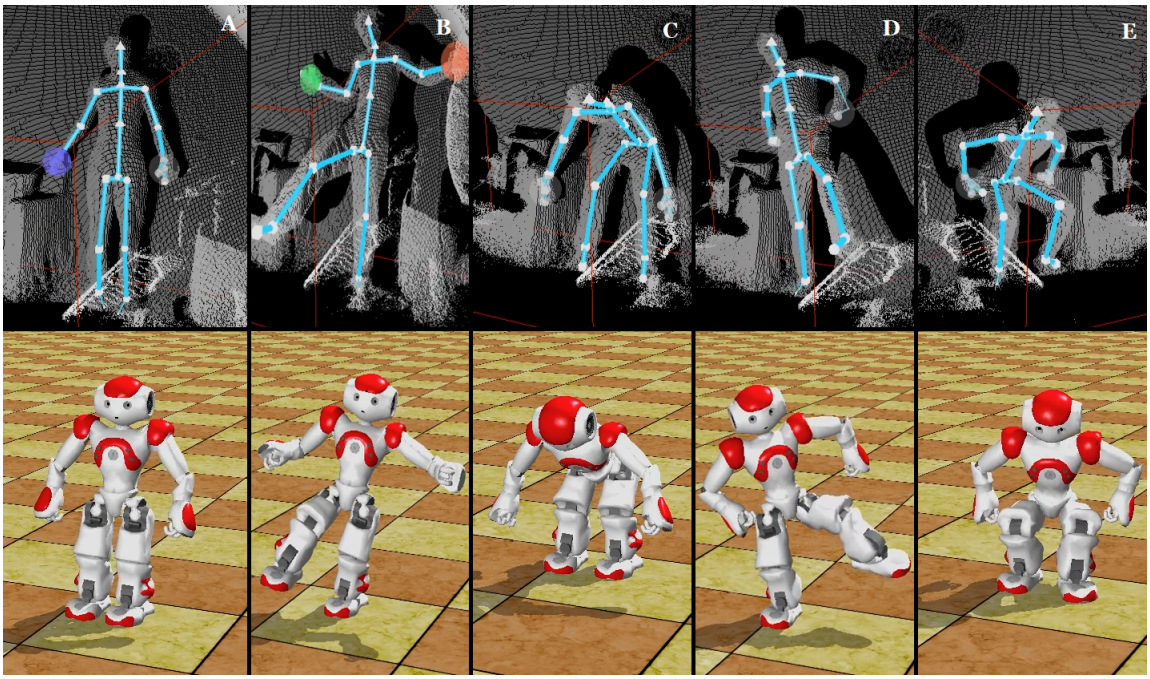

Figure 5.4: The five poses that have been selected to evaluate the effect of the balance controller.

and the center of mass and the right ankle and the center of mass is measured. Also, the minimal distance between the center of mass and the line between the two ankles is measured which is the distance from the center of mass to the optimal position in the double-foot support state. The coordinate system is projected onto the xz-plane, as the height of the center of mass does not influence the stability of the robot. The three measurements are performed in parallel on a robot with balance control and a robot without balance control. The results are shown in Fig. 5.5 and Fig. 5.6.

The graphs in Fig. 5.5 and Fig. 5.6 show the balance controller is effective. During the single-foot support phases, the balance controller successfully shifts the center of mass to the supporting foot by reducing the distance from the COM to the supporting ankle to zero. The graph that corresponds with the robot without balance control shows this distance was around 15 centimetres at the maximum during the performance of posture D in which the robot is in the right-foot support mode, which would make the robot fall when gravity is enabled as the center of mass lays outside of the support polygon. The graph in Fig. 5.5 shows the maximum minimal distance from the COM to the optimal location, the line between the two ankles, is only approximately five centimetres at the maximum when the balance controller is enabled. The corresponding maximum distance

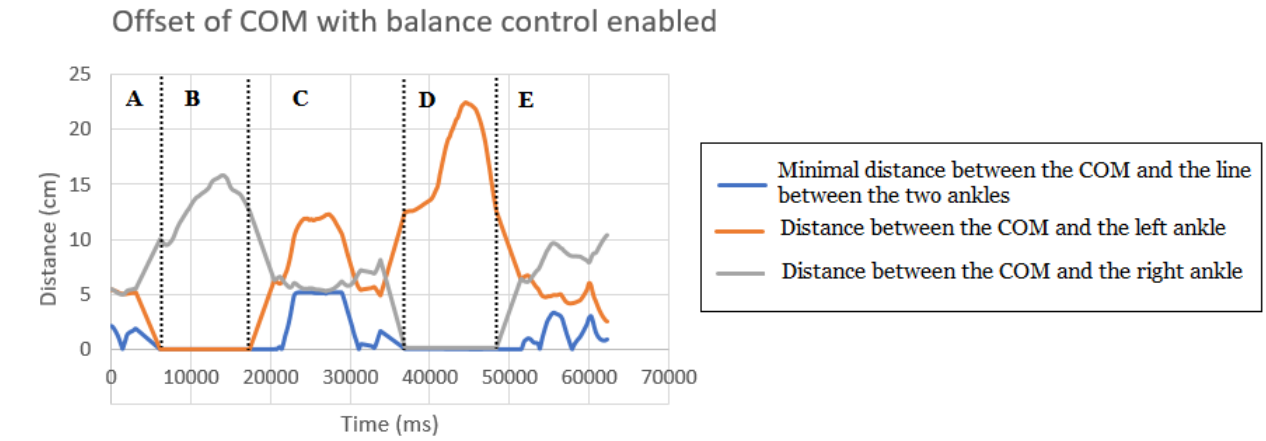

Figure 5.5: Distance between the COM and the line between the two ankles, the left ankle and the right ankle in centimetres with balance control enabled during the consecutive performance of the poses illustrated in Fig. 5.4. The coordinate system is projected onto the xz-plane.

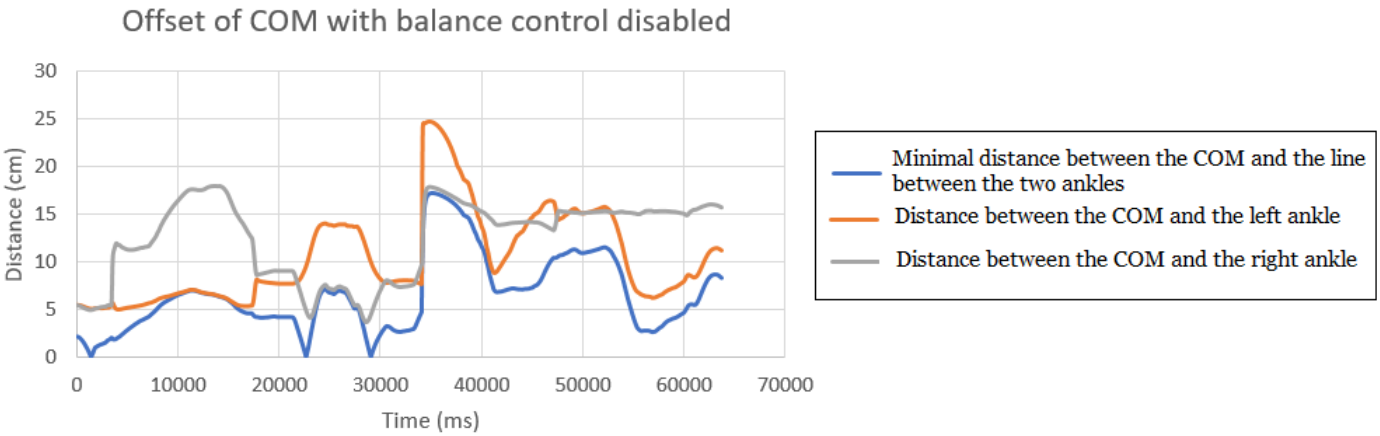

Figure 5.6: Distance between the COM and the line between the two ankles, the left ankle and the right ankle in centimetres with balance control disabled during the consecutive performance of the poses illustrated in Fig. 5.4. The coordinate system is projected onto the xz-plane.

of the robot with the balance controller disabled is circa 17 centimetres during the performance of posture C in which the robot is in the double-foot support mode. This would make the robot fall as the center of mass is located outside of the support polygon.

5.2 Experiments on the real robot

Several postures in the single-foot support phase and double-foot support phase have been performed by a real NAO robot. Four complex postures are shown in Fig. 5.7. Transitions between support modes could sometimes cause a fall of the robot. Putting the free foot back on the ground sometimes happens with a too high velocity, however this is done by using the NAOqi framework in which the velocity cannot be changed. Thus improvements can be made when putting a free foot back on the ground and when lifting a foot from

the ground to make the system more robust. When the robot has successfully switched from support mode it is able to imitate very complex postures, for example the fourth posture shown in Fig. 5.7.

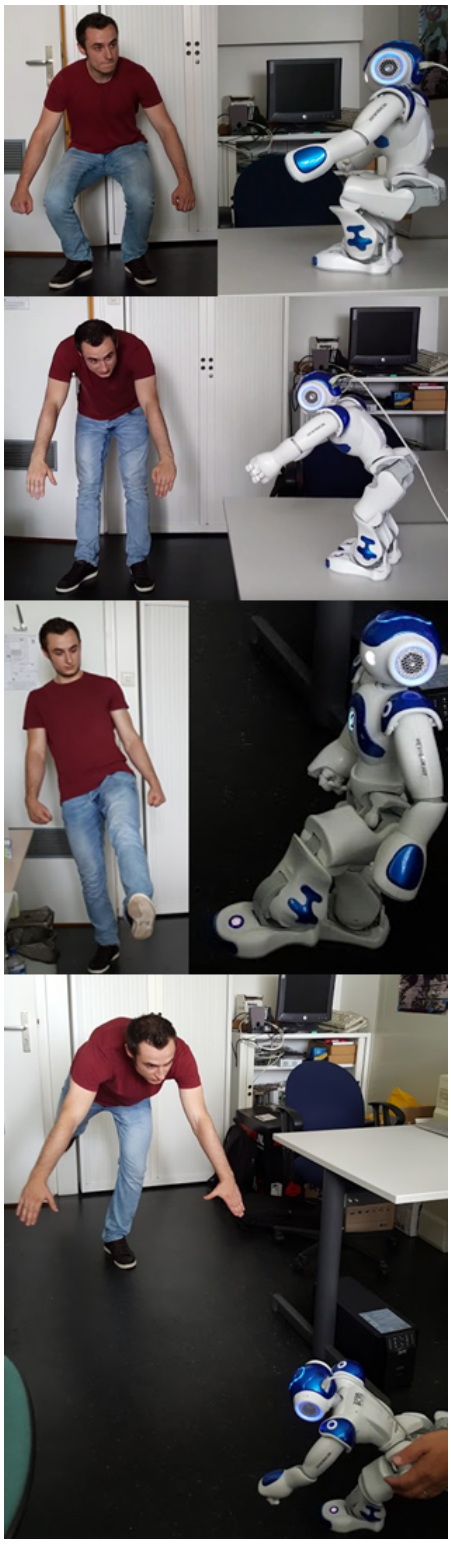

Figure 5.7

Chapter 6

Conclusion

In this paper a human motion imitation system for humanoids has been proposed. The imitation process consists of two parallel sequences. In the first sequence, human motions are captured with the Kinect 2 motion capture system. The coordinates of the human joints are extracted from the Kinect data. Finally the desired target angles of the robot corresponding to the human joint coordinates are calculated through Inverse Kinematics. This is done by minimizing an evaluation function which calculates the distance between the human posture and a robot posture. In the second sequence, small steps are made by the robot towards the target angles. For each such step, balance maintenance is applied with the use of the qpOases library. Experiments on a NAO robot have been performed. The experiments on the virtual NAO robot in simulator Webots show the robot closely follows the human motion. The angle trajectories of the limbs of the robot closely follow the angle trajectories of the limbs of the human over time with little delay. The delay is caused by maximum joint angle velocities, balance control modifications and computation time. In some cases an angle cannot be reached because it is outside of the range of motion of the robot. The imitation system also effectively reduces the offsets from the current joint location to the desired joint location over time. Experiments of the balance control show the balance control is effective as the distance from the center of mass to the desired location is significantly lower when balance control is enabled as compared to the case in which balance control is disabled. The experiments on a real NAO robot show the system is capable of imitating complex human motions in reality. Future work would be improving the balance maintenance of the robot during the transitions between support modes, as the transitions could sometimes cause a fall of the robot. A system that puts the free foot of the robot back on the ground at a lower pace could be implemented to make the transition of single-foot support state to double-foot support state more stable. To improve the lifting of the foot from the ground, a fixed motion could be implemented for the leg that has to be lifted from the ground, after which the normal imitation process is continued. Furthermore, code of other imitation systems could be requested to compare the systems by performing simultaneous experiments.

Bibliography

[1] Samuel R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*, 17, 2004.

[2] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

[3] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. `http://eigen.tuxfamily.org`, 2010 (accessed August 21, 2017).

[4] J. Koenemann, F. Burget, and M. Bennewitz. Real-time imitation of human whole-body motions by humanoids. *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[5] N. Kofinas. Forward and Inverse Kinematics for the NAO Humanoid Robot. Technical University of Crete, 2012.

[6] N. Kofinas. NAOKinematics. `https://github.com/kouretes/NAOKinematics`, 2012 (accessed August 21, 2017).

[7] J. Lei, M. Song, Z. Li, and C. Chen. Whole-body humanoid robot imitation with pose similarity evaluation. *Elsevier. Signal Processing*, 108:136–146, 2015.

[8] Y. Ou, J. Hu, Z. Wang, Y. Fu, X. Wu, and X. Li. A real-time human imitation system using Kinect. *International Journal of Social Robotics*, 7:587–600, 2015.

[9] B. Stephens. Integral control of humanoid balance. *Intelligent Robots and Systems, 2007.*, pages 4020–4027, 2007.