AUTOMATIC PLACEMENT OF THE HUMAN HEAD THANKS TO ERGONOMIC AND VISUAL CONSTRAINTS

Bilal Boualem, Damien Chablat, Abdelhak Moussaoui

Proceedings of the ASME 2015 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2015
August 2-5, 2015, Boston, Massachusetts, USA

DETC2015-46153

DRAFT: AUTOMATIC PLACEMENT OF THE HUMAN HEAD THANKS TO ERGONOMIC AND VISUAL CONSTRAINTS

Bilal BOUALEM
University of Tlemcen
Tlemcen, ALGERIA
Institut de Recherche en Communication et Cybernétique de Nantes, UMR CNRS 6597
bilal.boualem@ec-nantes.fr

Damien CHABLAT
Institut de Recherche en Communication et Cybernétique de Nantes, UMR CNRS 6597,
Nantes, France
Damien.Chablat@cnrs.fr

Abdelhak MOUSSAOUI
University of Tlemcen
Tlemcen, ALGERIA
University of Lorraine
Metz, FRANCE
abdelhak.moussaoui@univ-lorraine.fr

ABSTRACT

This article aims to create automatic placement and trajectory generation for the head and the eyes of a virtual mannequin. This feature allows the engineer using off-line programming software of mannequin to quickly verify the usability and accessibility of a visual task. An inverse kinematic model is developed taking into account the joint limits of the neck and the eyes as well as interference between the field of view and the environmental objects. This model uses the kinematic redundancy mechanism: the head and eyes. The resolution algorithm is presented in a planar case for educational reasons and in a spatial case.

INTRODUCTION

The need for designers, builders and manufacturers of tools that allow the study of the interaction of the humans with their environments and the designed products has prompted researchers to develop tools for the study of accessibility.

Computer tools (CATIA, SolidWorks, NX, PCT CREO, SANTOS, etc.); using virtual reality; have emerged to meet this demand. These tools are used to evaluate the mechanical and visual capabilities of the human to accomplish a task (Driving a car, using the computer keyboard, dial a phone number). However, there are a lot of drawbacks in these tools. For example, in the mannequin simulation, we are able to look at the hand but without taking into account the distance between the hand and the eyes and the size of the object.

In the literature, we find different approaches to compute the head motion to get a visible target. In [1], the author proposes an approach based on a multi-agent system that distributes the computation of the posture of the mannequin over several elementary agents (an agent to move the mannequin, an agent for the detection of collisions, an agent to keep the target visible, etc.). Another approach presented by T. Marler in [2] where he considers the vision as a criterion among several criteria to be satisfied. This is known as a multi-objective optimization problem. There are several methods and algorithms to solve it [3]. The works cited above define a cone of vision that describes the field of view of the eyes. An object is considered as visible if it belongs to this cone and no obstacle interferes between the object and the eyes. The distance between the eyes and the object, the size of the object are not considered.

In our approach, we consider the eyes as a part of the human upper body model, in other words, neck, head, eyes and vector of sight are considered as an additional end of the body of the mannequin. This end of the body is modeled according to the capabilities of the eyes.

The visibility problem becomes an inverse kinematics problem. We use the method described by Aristido in **[4]** to solve it. We improve this method to detect and avoid obstacles present inside the environment.

HUMAN UPPER BODY MODELING

In this section, we describe the geometrical model of the human upper body. We start by the presentation of the model proposed by T. Marler in [2]. The author proposes a model with 26 degrees of freedom (dof) as it is shown in the Fig. 1: 12 dof are used to model the spine, 2 dof for the collarbone and 3 for the shoulder, 2 for the elbow, 2 for the wrist and 5 dof for the

neck. We notice that all joints are rotational. We use this model because it gives a realistic posture of the mannequin with a reduced computational cost.

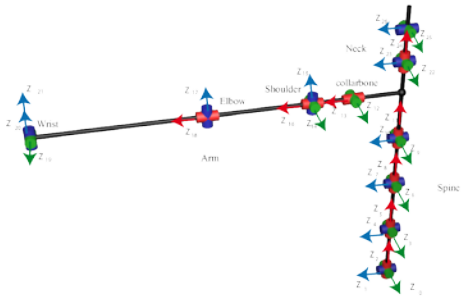

Fig. 1: T. Marler's Human upper body Model

In our approach, we consider the eye features as an extension of the human upper body model. In other words, the eyes features are an additional joints and links added to the neck model.

The ability for human to see an object is related to the eyes features (Eyes motion, field of view, and visual acuity) and the environment (lighting, obstacle between the eyes and the object). In this work, we are not interested by the lighting conditions.

Visual acuity

The visual acuity is the ability to distinguish the details of the object. According to this report [5], there are several techniques to assess the visual acuity. The most used are the Snellen's chart described by I. Bailey in [6]. The technique is based on the ability of the person to read a letter of a size S from a certain distance D. The Figure 2, shows the Snellen principle, where S is the height of the letter (or object) and D is the distance between the eyes and the object, α is the angular size of the letter in minutes of arc (1 minute of arc = π/10800 radians).

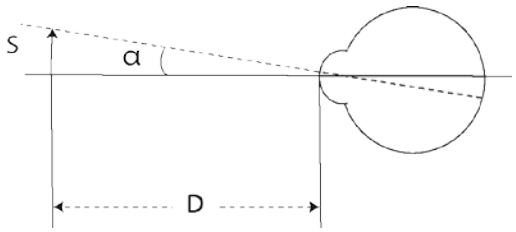

Fig. 2: Snellen principle

The Fig. 3 shows an example of a letter used to assess the visual acuity. The letter is composed of three black zones and two white zones. A human can read this letter if he can distinguish every zone.

Fig. 3: Snellen's letter

The size of every zone (α/5) is called the minimum angle of resolution (MAR). Using the MAR, we can get the maximum distance to see an object.

$$D_{max} = \frac{S}{tan\left(5MAR_{far}\right)} \quad (1)$$

where MAR_{far} is the MAR for far vison.

In the same way, we use the MAR for the near vision to compute the minimum distance [7].

$$D_{min} = \frac{S}{tan\left(5MAR_{near}\right)} \quad (2)$$

where MAR_{near} is the MAR for near vison.

The object must be at a distance $D \in \left[D_{min},\ D_{max}\right]$ to be visible. This can be modelled as a prismatic joint that gives the ability to vary the length of the virtual link (the look vector).

The field of view

The field of view is the visible space for a motionless eyes and head. According to Parker [8], the field of view is the union of the visible spaces of the two eyes, as it is shown in the Fig. 4. The blue zone represents the right eye and the red zone the left eye. The intersection zone, called the binocular vision. The human solicits both eyes to see.

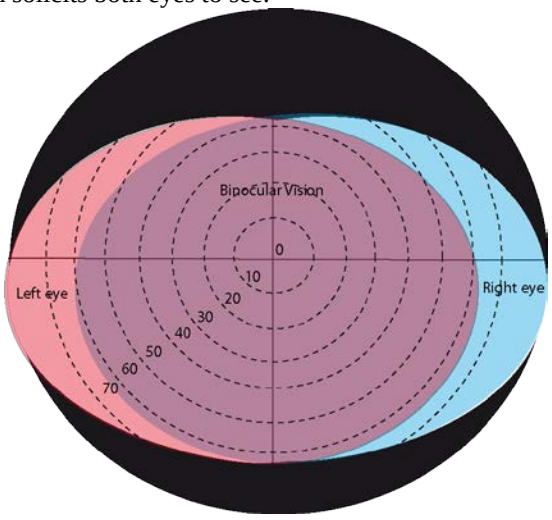

Fig. 4: Field of view

The visual acuity is at the maximum in the centre zone, called the fovea. The visual acuity drops exponentially when the

object is far from the centre. In [2]. The author proposes a function to compute the deterioration of the acuity visual according to the angle between the eye to target vector and the fovea. It is expressed as:

$$\beta = e^{-7|\theta|} \quad (3)$$

where θ represents the angle between the eye to target vector and the fovea vector.

Therefore, the problem is to reduce θ. Considering the look vector (fovea) as an virtual link, The application of an inverse kinematics algorithm resolve the problem, if the virtual effector reach the target $\theta = 0$, if not the algorithm try to close to the target. At this moment, we can check the visibility of the target by reducing the visual acuity using (3). Then we compute the minimal and maximal distance using (1) and (2). Finally, check if the distance eye to target belongs to $[D_{min}, D_{max}]$.

The eyes movements

The aim of moving the eyes is to enclose the object to the fovea zone.

According to [9] and [10], the eyes movements can be represented using two rotations. The first rotation moves the gaze up and down while the second one moves it right and left. The range of the motion is $\pm 55°$.

We can conclude that the eyes behave as a joint with 2 dof. Therefore, to model the eyes movements we add a joint with 2 dof on the top of the neck; this joint orients a virtual link (that represents the look vector) to the target.

The Full model

Figure 5 shows the model of the human upper body considering the eyes features. The eyes are modelled using a prismatic joint representing the visual acuity and a rotational joint with 2 dof to represent the eyes motions.

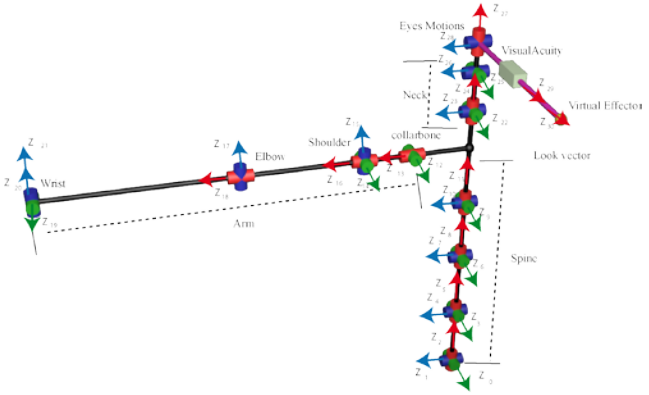

Fig. 5: The human upper body

Using this approach to model the eyes behaviour, we can define different targets for the hands and the eyes. It is possible, also, to give more importance to the reachability of the hand or the visibility.

FORMULATION OF THE PROBLEM

Now, the problem is to find the posture of the mannequin that allows to the two arms to reach the targets. The problem is known in the literature as an inverse kinematics problem it can be formulated as:

We assume that the current positions of the effectors are P_{eff1} and P_{eff2} and the current orientations are $R_{eff1} = [X_{eff1}, Y_{eff1}, Z_{eff1}]$ and $R_{eff2} = [X_{eff2}, Y_{eff2}, Z_{eff2}]$ (where X_{eff}, Y_{eff} and Z_{eff} are the unit vectors associated to the effector).

Given a desired positions t_1 and t_2 and the desired orientations are $R_{d1} = [X_{d1}, Y_{d1}, Z_{d1}]$ and $R_{d2} = [X_{d2}, Y_{d2}, Z_{d2}]$.

Find the posture such as:

$$P_{eff1} = t_1$$
$$P_{eff2} = t_2$$
$$R_{eff1} = R_{d1}$$
$$R_{eff2} = R_{d2}$$

INVERSE KINEMATICS ALGORITHM

The cyclic coordinate descent (CCD) algorithm, described in [11], used in many fields to solve the IK problem (i.e compute the posture of an articulated body), this method considered as a fast and robust algorithm but the posture obtained is unrealistic making it not adapted for animating human.

A. Aristidou proposes an algorithm, called Forward and backward invers kinematics (FABRIK) in [4], to solve IK problem. This algorithm is faster, robust and offers a realistic posture. In addition, the geometrical aspect of the algorithm makes the detection and the avoidance of obstacles simpler. These advantages justify our choice to use this algorithm.

In this section, we describe the Forward and backward invers kinematics algorithm, developed by A. Aristidou. This algorithm is very fast, and the solution obtained is realistic, compared to other IK algorithms (CCD).

We take the example of the manipulator, shown in Fig. 6 to explain FABRIK algorithm. The manipulator contains four joints connected by four links. We note P_i the Cartesian position of the joint "*i*" and d_i the length of the link "*i*". We can distinguish two cases:

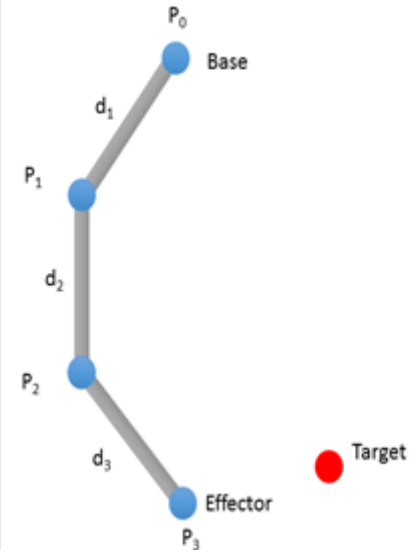

Fig. 6: An example of a planar manipulator

— **Unconstrained case:** FABRIK method is an iterative algorithm that solves the IK problem, each iteration of the algorithm divided on two steps:

On the first loop, called forward step, we place the effector on the target position then we move the n-1th joint on the line linking the effector and P_{n-1}, We note the new position P'_{n-1}. P'_{n-1} is at d_{n-1} of the effector as it is shown in the Figure 7-1. In the same way, we move the next joints until the base (P_0), as shown in Fig. 7-2 and 7-3.

However, in the backward step, we start by placing the base (first joint of the manipulator) on its original position, and then we move the joint 2 on the line connecting the original position of the base and the P'_2, we place P_2 at d_1 from the base. Repeating the iteration (Forward and backward steps), cancels the error between the effector and the target.

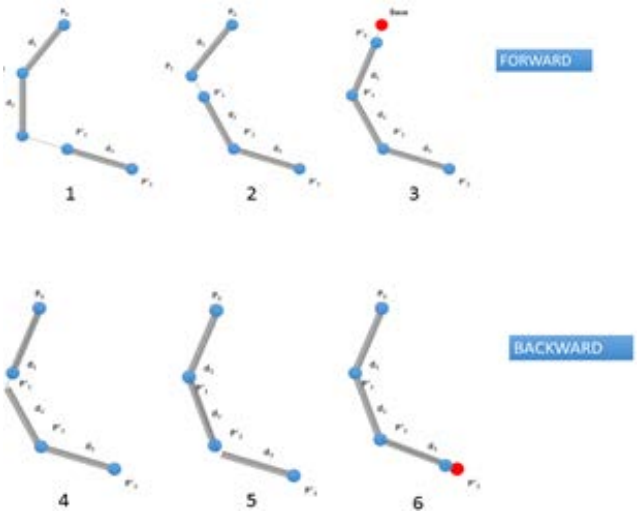

Fig. 7: FABRIK Sequence

The algorithm is fast, simple to implement and the posture obtained is more realistic then the other IK method. This makes this algorithm the best compromise in the unconstrained case.

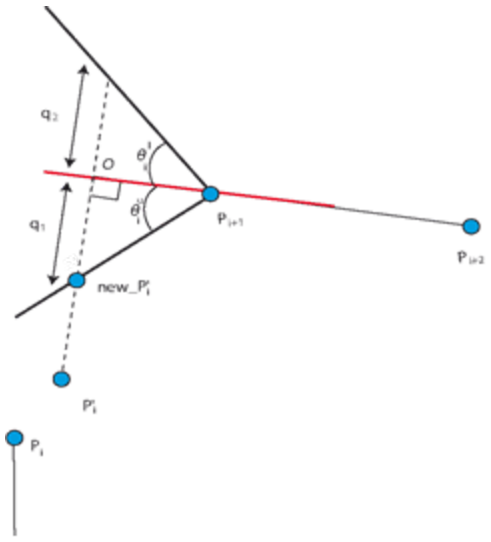

Fig. 8: Constrained FABRIK

— **Constrained case:** In this case, the constraints represent the joint limits; they are defined on the joint space. It is hard to consider the joints limits in the compute steps because of the difference between the Cartesian space and the joints space. A. Aristidou proposes a method to respect joints limits. The goal of this method is to check if the computed position of the joint i (on the forward or backward loop) P'_i is in the reachable set of the joint else take the closest point to P'_i. The Figure 8 shows a planar example to check if the position P'_i respecting the joint limits (θ^l_i, θ^u_i). First, we must define the reachable set. So, we starts by compute the projection, noted O, of the point P'_i on the extension of the segment P_{i+1} P_{i+2} (red line) as shown in the Figure 8. Then we compute $q_1 = || \mathrm{O} - \mathrm{P}_{i-1} || \, tan(\theta^u_i)$ and $q_2 = || \mathrm{O} - \mathrm{P}_{i-1} || \, tan(\theta^l_i)$, then we check if P'_i is in the segment $[O - q_2, O + q_1]$, then keep P'_i. Else we choose the nearest position to P'_i (new_ P'_i) as it is shown in the Figure 8. In the 3D case, A. Aristidou defines the reachable set as an ellipse centred on O and $q_1 + q_2$ and $q_3 + q_4$ are major and minor radius.

This method work perfectly especially when the joints limits are lower than $\boldsymbol{\pi}/2$, but it is unstable when the boundaries are equal to $\boldsymbol{\pi}/2$; this instability is due to the tangent function.

To respect the joints limits, it is necessary to find a transformation that allows the transition from the joint space to the Cartesian space or the inverse (i.e from the Cartesian space to the joint space). The author chosen to use tangent function to do the transformation but this function presents singularity at π/2 that make the algorithm unstable near this singularity. Therefore, we need a reliable and a fast algorithm to respect joint limits.

In the next section, we propose a new reliable method to respect the joints limits.

The Figure 9 shows a manipulator in 3D space. This manipulator is composed of four rotational joints. We call X_i the unit vector associated to the extension of the segment [P_{i-1} P_i] (or [P_{i+1} P_i] in the forward step) illustrated by the red vector in the Figure 9. Y_i, Z_i (blue and green vectors) are a units vectors belonging to the plan perpendicular to the vector X_i as it is shown in the Figure 9. We call system "i", the frame composed of (X_i , Y_i, Z_i).

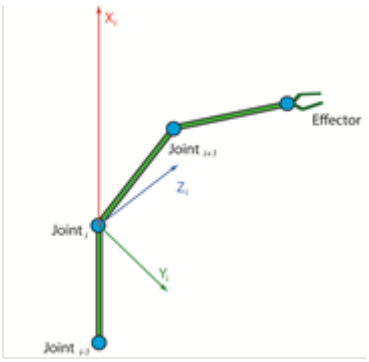

Fig. 9: An articulated body in 3D

Formulation of the problem

The problem can be defined as:

Given P'_i the new position of the joint "*i*" computed by FABRIK, and [θ^l_i, θ^u_i] is the lower and the upper limits of the joints "i". Considering a joint limit restricts the set of reachable points (SRP).

The problem is to check if the point P'_i belongs in the SRP to the joint "i". Then keep P'_i.

Else take the closest point to P'_i from SRP.

Constrained FABRIK

To apply the joint limits to the joint "*i*", we divide into two problems, the first is the orientation problem; it consists to orients the Y_i, Z_i vectors. The second is the position problem. It consists to check if the new position of the joint "i" P'_i respects the joint constraints.

Assume we are in the first step (i.e forward loop), to apply the joint limits, we start by orienting (X_i, Y_i, Z_i) such that the segment $P_{i+1}P_i$ and *Xi* become collinear. For this, we need to find the rotor expressing the rotation between the X_i vector and the segment $P_{i+1}P_i$. We use the quaternion to represent the rotation around a vector, because it is simpler, faster and more efficient than the rotation matrix, according to [12].

To find the quaternion we need to find the rotation vector V and the angle α between the two vectors, X_i and $P_{i+1}P_i$. V is the normal vector perpendicular to the plane (X_i, $P_{i+1}P_i$). We note that the use of P'_i or P_i to compute the vector V and the angle α is the same because P'_i, P_i and P_{i+1} are in the same line.

$$V = Xi \wedge \left(\frac{P_{i+1} - P_i}{\left\| P_{i+1} - P_i \right\|} \right)$$

and the angle can be found using the dot product as follow:

$$\alpha = ar\cos\left(Xi \bullet \left(\frac{P_{i+1} - P_i}{\left\| P_{i+1} - P_i \right\|} \right) \right)$$

Now, we define the quaternion such as

$$Q = \left[cos\left(\frac{\alpha}{2}\right); sin\left(\frac{\alpha}{2}\right) V \right]$$

Using the quaternion, we compute the new vectors X'_i, Y'_i and Z'_i by rotating each vector using the quaternion as follow:

$$Y'_i = QVQ^{-1}$$

Where $Q^{-1} = Q' / \left\| Q \right\|$, $Q' = \left[cos\left(\frac{\alpha}{2}\right); -sin\left(\frac{\alpha}{2}\right) V \right]$ is the conjugate of Q.

The multiplication of two quaternions $\mathrm{p} = \left[p_1 p_2 p_3 p_4 \right]$ and $\mathrm{q} = \left[q_1 q_2 q_3 q_4 \right]$ is

$$\mathrm{pq} = \begin{bmatrix} p_1q_1 - p_2q_2 - p_3q_3 - p_4q_4 \\ p_1q_2 + p_2q_1 + p_3q_4 - p_4q_3 \\ p_1q_3 - p_2q_4 + p_3q_1 + p_4q_2 \\ p_1q_4 + p_2q_3 - p_3q_2 + p_4q_1 \end{bmatrix}$$

The orientation constraints

In the forward step, Given the two systems: "i+1" (X_{i+1}, Y_{i+1}, Z_{i+1}) associated to the joint "i+1" and "i" (X_i, Y_i, Z_i) associated to the joint "i".

If X_i and X_{i+1} are collinear then the orientation of the system "*i*" is the angle between (Y_i and Y_{i+1}) or (Z_i and Z_{i+1}).

So, considering the system "*i+1*" is fixed, the problem is to find the angle θ_1 between the vectors (Y_i and Y_{i+1}) or (Z_i and Z_{i+1}). θ_1 must respects the orientation:

$$\theta^l_i < \theta_1 < \theta^u_i$$

If X_i and X_{i+1} are not collinear, we need to express the rotor such that they become collinear. Using this rotor, we compute

the orientation angle. It is possible to use a decomposition to the Euler angles as described in [12].

To compute the rotor, we start by projecting X_i on the (X_{i+1} Y_{i+1}) and (X_{i+1} Z_{i+1}) planes, using the algorithm presented on [13]. Using the cross and the dot product it is easy to find the rotations φ and ψ around the axes Y_{i+1} and Z_{i+1} to have the vetors X_i, X_{i+1} collinear.

The rotation angle θ_1 represents the angle between (Y_i and Y_{i+1}) or (Z_i *and* Z_{i+1}) as long as the two vectors X_i, X_{i+1} are collinear (we must rotate Y_i to be in the Y_{i+1} Z_{i+1} plane).

Therefore, using the quaternion, we rotate Y_i around the vectors Y_{i+1} and Z_{i+1} to find Y'_i, the rotation result of Y_i. This transformation allows us to compute the angle:

$$\theta_1 = ar\cos\left(Y_i' \bullet Y_{i+1}\right)$$

Now, we can check if

$$\theta^l{}_i < \theta_1 < \theta^u{}_i$$

If $\theta_1 \in [\theta^l_i, \theta^u_i]$ then keep θ_1. However, if $\theta_1 \notin [\theta^l_i, \theta^u_i]$, we take the closest boundary θ^l_i, θ^u_i. Then we compute the new Y_i and Z_i by rotate Y_{i+1} and Z_{i+1} around X_{i+1}, Y_{i+1} and Z_{i+1} by new_θ, ψ *and* φ respectively.

Algorithm

Inputs: $[X_i, Y_i, Z_i]$, $[X_{i+1}, Y_{i+1}, Z_{i+1}]$ and θ^l_i, θ^u_i
Outputs: $[X_i, Y_i, Z_i]$

- *Compute* φ *and* ψ *by projecting* X_i *on the* $X_{i+1}Y_{i+1}$ *and* $X_{i+1}Z_{i+1}$ *plans*
- Q_1= *[cos* (ψ /2); *sin* (ψ /2) Y_{i+1}]
- Q_2= *[cos* (φ /2); *sin* (φ /2) Z_{i+1}]
- $Y'_i = Q_1 * Q_2 * Y_i * Q_1^{-1} * Q_2^{-1}$
- θ_1 = *acos* (<$Y'i$, Y_{i+1}>)
- **If** $\theta_1 \in [\theta^l_i, \theta^u_i]$ **then** keep θ_1.
- **Else**
  if $(\theta_1 - \theta^l_i)^2 < (\theta_1 - \theta^u_i)^2$ **then** new_$\theta_1 = \theta^l_i$
  Else new_$\theta_1 = \theta^u_i$
- Q_1= *[cos* (ψ /2); *sin* (ψ /2) Y_{i+1}]
- Q_2= *[cos* (φ /2); *sin* (φ /2) Z_{i+1}]
- Q_2= *[cos* (new_θ_1 /2); *sin* (new_θ_1 /2) X_{i+1}]
- $Q = Q_1 * Q_2 * Q_2$
- Y_i =Q * Y_{i+1} * Q^{-1}
- Z_i = Q * Z_{i+1} * Q^{-1}

The position constraints

P'_i is a combination of the rotation around X_i, Y_i and Z_i vectors, in the previous section we described a method for computing the orientation of Y_i and Z_i vectors, Now, we are interested in calculating the rotation angles around Y_i and Z_i vectors. Instead of considering the 3D Problem, we divide it into two 2D problems. This will simplify it.

We note that a rotation around Y_i axis only, generates a variation of P'_i in the (X_i Z_i) planar and a rotation around Z_i axis only generates a variation of P'_i in the (X_i Y_i) planar. Therefore, to find the rotation angles, we simply project P'_i on the two planes (X_i Y_i) and (X_i Z_i). We note P'_{jz}, P'_{jy} the projections of P'_i on the X_iZ_i, X_iY_i planes respectively.

Now, It is possible to find the angles θ_2 (between the segment $[P'_{i+1}P'_{jz}]$ and X_i) and θ_3 (between the segment $[P'_{i+1}P'_{jy}]$, X_i), using the dot and the cross products.

After that, we can check if

$$\theta^l{}_2 < \theta_2 < \theta^u{}_2$$

$$\theta^l{}_3 < \theta_3 < \theta^u{}_3$$

If $\theta_i \in [\theta^l_i, \theta^u_i]$ then keep θ_i. However, if $\theta_i \notin [\theta^l_i, \theta^u_i]$, we take the closest boundary θ^l_i, θ^u_i. We note that to check θ_2 (the angle between the segment $P'_{i+1}P'_{jz}$ and X_i axis) we use the limits applied to the rotation around Y_i axis.

We compute the new P'_{jy} P'_{jz}, by using the following expressions:

$$P'_{jy} = cos\left(new\_\theta_3\right)\left(cos\left(new\_\theta_2\right)x_i + sin\left(new\_\theta_2\right)y_i\right)$$

$$P'_{jz} = cos\left(new\_\theta_2\right)\left(cos\left(new\_\theta_3\right)x_i + sin\left(new\_\theta_3\right)z_i\right)$$

Now, we need to compute the new position P'_i from the projections P'_{jy}, P'_{jz}. Therefore, P'_i is the point of intersection between the normal vectors of (X_iY_i) and (X_iZ_i) planes on the projection point P'_{jy}, P'_{jz}.

Since the points P'_i, P_i and P_{i+1} are on the same line it is possible to use P_i directly to check the position constraints without computing the new position P'_i. It will be computed at the end of the position constraint algorithm.

Algorithm

Inputs: P_i, P_{i+1}, $[X_i, Y_i, Z_i]$, *di* and θ^l_i, θ^u_i
Outputs: P'_i

- *Compute* P'_{jy} *and* P'_{jz}, *the projections of Pi on the* X_iY_i *and* X_iZ_i *plans*
- θ_2 = *acos* (<Xi , (P'_{jz} - P_{i+1})>/|| (P'_{jz} - P_{i+1})||)
- θ_3 = *acos* (<Xi , (P'_{jy} - P_{i+1})>/|| (P'_{jy} - P_{i+1})||)
- *check* $\theta^l_i < \theta_2 < \theta^u_i$
  $\theta^l_i < \theta_3 < \theta^u_i$
- P'_{jy} = *cos* (new_θ_3) (*cos* (new_θ_2) x_i + *sin* (new_θ_2) y_i)
- P'_{jz} = *cos* (new_θ_2) (*cos* (new_θ_3) x_i + *sin* (new_θ_3) z_i)
- n_1=*cross*(X_i , Y_i)
- n_2= *cross*(X_i , Z_i)
- *Find new_P'i the intersection between* n_1 *and* n_2 *at* P'_{jy} , P'_{jz}.
- P'_i =*di* (new_P'_i - P_{i+1})/|| (new_P'_i - P_{i+1}) ||

The full algorithm

Assuming, we have *n* dof manipulator and P_i and $[X_i, Y_i, Z_i]$ are the positions and the coordinates systems associated to the joint "*i*", we pose *t* and $R_t = [X_t, Y_t, Z_t]$ the desired position and orientation of the effector, respectively.

The consideration of the joint limits will affect the speed of the algorithm, but we can simplify the algorithm. The full algorithm becomes

Algorithm

Inputs: $t, R_t, P_i, [X_i, Y_i, Z_i]$ and θ^l_i, θ^u_i
Outputs: $P_i, [X_i, Y_i, Z_i]$

- $\varepsilon = 0.1$
- $di = \| P_i - P_{i+1}\|$ $i=1,\ldots,n$
- **if** $\| P_i - P_{i+1}\| > d_1 + d_2 + \ldots + d_{n-1}$
  then *t is unreachable*
- **else**

  - $b = P_1$ % keep the base position
  - $[X_{base}, Y_{base}, Z_{base}] = [X_1, Y_1, Z_1]$ % keep the units vectors of the base (joint 1)
  - $error = \| t - P_n\|$ Compute the error between the target t and the effector
  - ***while*** $error > \varepsilon$

    % forward step
    $P_n = t$
    $[X_n, Y_n, Z_n] = R_t$
    For $i=n-1,\ldots,1$
    % call the position constraint function
    P_i =contrainte_position(P_i, P_{i+1}, $[X_{i+1}, Y_{i+1}, Z_{i+1}]$, $[\theta^l_i \ \theta^u_i]$)
    %orienting the system
    $X_t = P_{i+1} - P_i / \| P_{i+1} - P_i \|$
    $Angle = acos(< X_i , X_t >)$
    $Axis = cross(X_i , X_t)$
    $[X_i, Y_i, Z_i]$=rotate($[X_i, Y_i, Z_i]$, Angle, Axis)
    %Orientation limits of the next coordinate system
    $[X_i, Y_i, Z_i]$ =constaint_orientation($[X_i, Y_i, Z_i]$, $[X_{i+1}, Y_{i+1}, Z_{i+1}]$, $[\theta^l_i \ \theta^u_i]$)
    End

    % backward step
    $P_1 = b$
    $[X_1, Y_1, Z_1] = [X_{base}, Y_{base}, Z_{base}]$

    For $i=1,\ldots,n-1$
    P_{i+1} =contrainte_position(P_{i+1}, P_i, $[X_{i+1}, Y_{i+1}, Z_{i+1}]$, $[\theta^l_i \ \theta^u_i]$)
    $X_t = P_{i+1} - P_i / \| P_{i+1} - P_i \|$
    $Angle = acos(< X_{i+1} , X_t >)$
    $Axis = cross(X_{i+1} , X_t)$
    $[X_{i+1}, Y_{i+1}, Z_{i+1}]$=rotate($[X_{i+1}, Y_{i+1}, Z_{i+1}]$, Angle, Axis)

    $[X_{i+1}, Y_{i+1}, Z_{i+1}]$ =constaint_orientation($[X_{i+1}, Y_{i+1}, Z_{i+1}]$, $[X_i, Y_i, Z_i]$, $[\theta^l_i \ \theta^u_i]$)
    End

  - $error = \| t - P_n\|$
  - ***End***

Spatial Constraints (Collision-free)

The mannequin had to avoid the collision with the objects present in the virtual environment while doing a task. We need to simplify the geometry of the objects to reduce the computations. There are lot of techniques of the simplification of the geometry used in virtual reality; the most used is the minimal enclosing box described in [14]. This technique simplifies the geometry of an object to a box. Another technique, presented by Hariri in [15], decomposes the object to several simple objects (spheres, planes, boxes), this technique is more accurate.

The links of the mannequin are considered as segments.

At each iteration "*i*" after computing the new position of the articulation P'_i (in the forward loop); we check the intersection between the segment $[P_{i+1}, P'_i]$ (in the forward loop) and the box; the existence of the intersection is the proof of a collision between the link and the object. The new position $new\_P'_i$ (that avoid the collision) need to guarantee three conditions:

1. No intersection between $[P_{i+1}; new\_P'_i]$ and the box
   This condition allows to avoid the obstacle

2. No intersection between $[new\_P'_i, P_{i-1}]$ and the box: This condition guarantees the existence of at least one solution in the next iteration. In the unconstrained case (without orientation and position constraints), this condition avoid to check the collision in the next iteration.

3. $new\_P'_i$ is the nearest position of the joint "*i*" to P'_i satisfying the previous conditions.

To find $new\_P'_i$, we discretize the SRP of the joint "*i*" (the positions of the joint "*i*" respecting the position constraints) as shown in Fig 10.

Among this set, the green arc defines the positions respecting the conditions (1 and 2); the pink part defines the positions respecting the first condition only and the red part defines the positions respecting any conditions from (1 and 2).

We choose the nearest position to P'_i that satisfies the conditions described above.

In constrained case (with orientation and position constraints), if there is any position of SRP that satisfied the first condition, we vary the previous joint to find the position that satisfy the conditions.

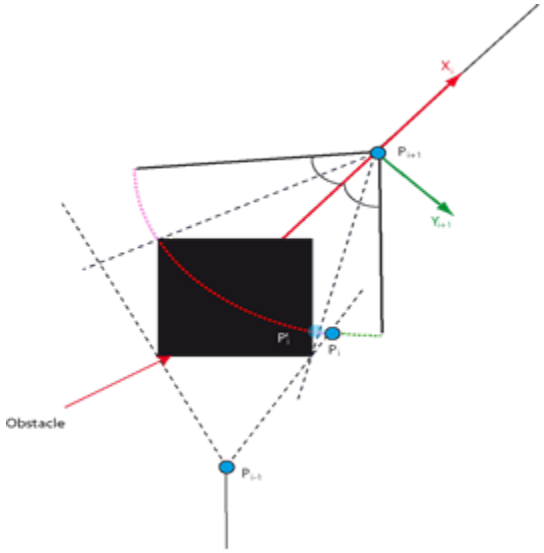

Fig. 10: Collision-free Approach

RESULTS AND DISCUSSION

To test the inverse kinematics algorithm considering joints and spatial constraints, we use an articulated body model with four rotational joints allowing a 3 DOF constrained motion. The articulated body is described in the Fig. 11.

Figure 11 shows an obstacle in the middle of the scene. The initial posture of the articulated body is shown in blue the target is the red "+ symbol". Figure 11 shows the evolution of the posture from the initial posture to the final one.

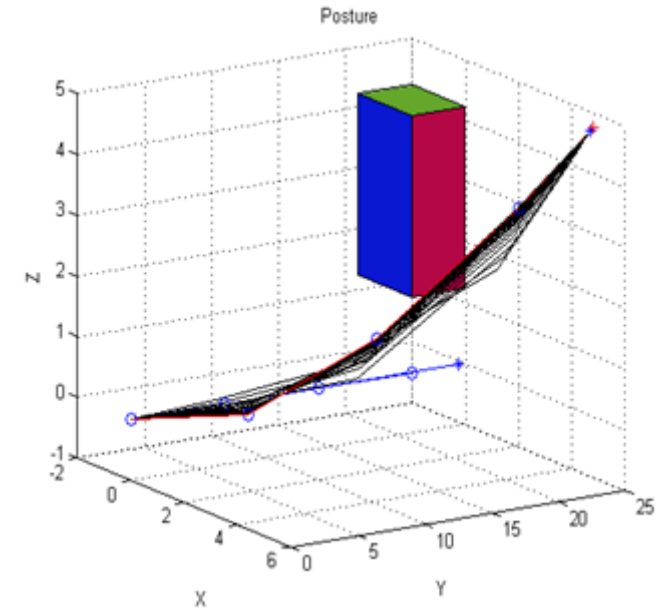

Fig. 11: Collision-free FABRIK results

We notice that the algorithm choose the best way to avoid the obstacle from the first posture. The final posture satisfies the orientation, position and the spatial constraints.

The error evolution is shown in the Figure 12. After 18 iterations, the algorithm converges to the solution.

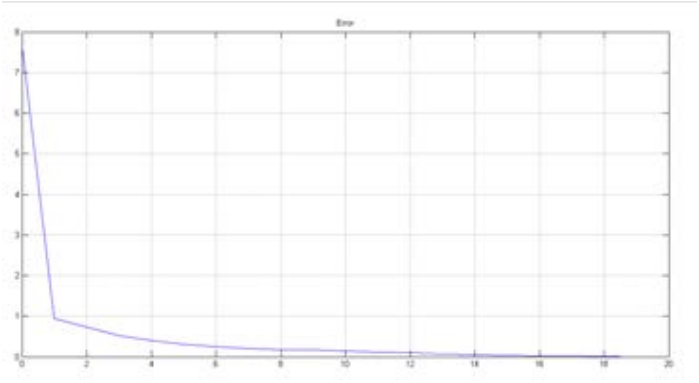

Fig. 12: Error evolution

Figure 13 shows the initial posture of the human upper body. The red "+ symbol" define the target of the eyes and the hand. The blue articulated body represents the spine this part is shared between the left arm (red) and the neck and eyes features part (green).

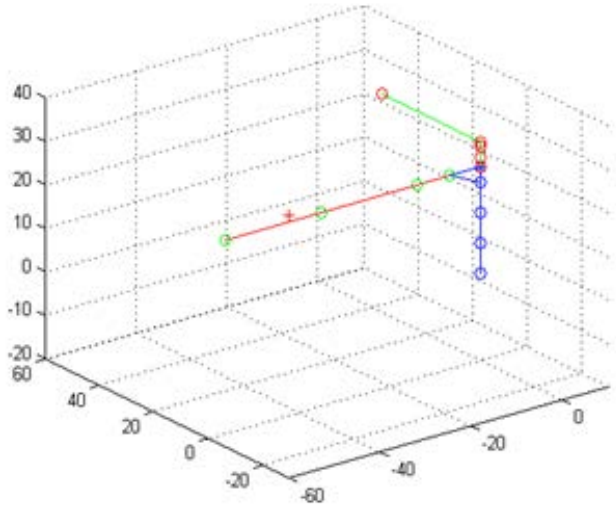

Fig. 13: initial posture

Applying the algorithm, we get the result shown in the Figure 14. We can notice that the vision influences the posture and specially the head position.

The algorithm is simple to implement and the posture is realistic. The speed of the algorithm depends essentially to the size of the obstacles and the technique chosen to simplify the geometry. These advantages are making this algorithm interesting in the virtual reality field.

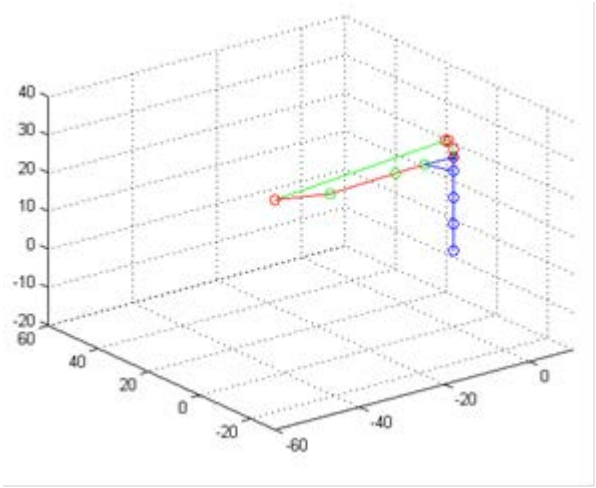

Fig. 14: Final posture

CONCLUSIONS

In this paper, we presented a new approach to model a human vision. It is based on the features of the eyes (Vision field, Visual acuity, eyes motion and no interference with obstacles). We consider the look vector as an additional end of the human model.

This modelling allows us to formulate the visibility problem as a classic IK problem for a multi arms articulated body.

We used an improved FABRIK algorithm to solve the IK problem for multi arms articulated body and we presented an algorithm to avoid the collision with obstacles. After detecting a collision between a link and an object, we try to find a position that avoids the collision by analysing the joint space.

For future works, we project to use different techniques to analyse the whole space [16] and considering a global solution of the IK problem avoiding collision.

ACKNOWLEDGMENTS

The work presented in this paper was partially funded by the Erasmus Mundus Active.

REFERENCES

[1] D. Chablat, P. Chedmail, and L. Pino, “A distributed approach for access and visibility task under ergonomic constraints with a manikin in a virtual reality environment,” in *10th IEEE International Workshop on Robot and Human Interactive Communication, 2001. Proceedings*, 2001, pp. 32–37.

[2] T. Marler, K. Farrell, J. Kim, S. Rahmatalla, and K. Abdel-Malek, “Vision Performance Measures for Optimization-Based Posture Prediction,” SAE International, Warrendale, PA, SAE Technical Paper 2006-01-2334, Jul. 2006.

[3] R. T. Marler and J. S. Arora, “Survey of multi-objective optimization methods for engineering,” *Struct. Multidiscip. Optim.*, vol. 26, no. 6, pp. 369–395, Mar. 2004.

[4] A. Aristidou and J. Lasenby, “FABRIK: A fast, iterative solver for the Inverse Kinematics problem,” *Graph. Models*, vol. 73, no. 5, pp. 243–260, Sep. 2011.

[5] “Recommended stardard procedures for the clinical measurement and specification of visual acuity. Report of working group 39. Committee on vision. Assembly of Behavioral and Social Sciences, National Research Council, National Academy of Sciences, Washington, D.C,” *Adv. Ophthalmol. Fortschritte Augenheilkd. Prog. En Ophtalmol.*, vol. 41, pp. 103–148, 1980.

[6] I. L. Bailey and J. E. Lovie, “New design principles for visual acuity letter charts,” *Am. J. Optom. Physiol. Opt.*, vol. 53, no. 11, pp. 740–745, Nov. 1976.

[7] I. L. Bailey and J. E. Lovie, “The design and use of a new near-vision chart,” *Am. J. Optom. Physiol. Opt.*, vol. 57, no. 6, pp. 378–387, Jun. 1980.

[8] J. F. Parker Jr. and V. R. West, “Bioastronautics Data Book: Second Edition. NASA SP-3006.,” *NASA Spec. Publ.*, vol. 3006, 1973.

[9] H. Misslisch, D. Tweed, and T. Vilis, “Neural Constraints on Eye Motion in Human Eye-Head Saccades,” *J. Neurophysiol.*, vol. 79, no. 2, pp. 859–869, Feb. 1998.

[10] D. Guitton and M. Volle, “Gaze control in humans: eye-head coordination during orienting movements to targets within and beyond the oculomotor range,” *J. Neurophysiol.*, vol. 58, no. 3, pp. 427–459, Sep. 1987.

[11] L.-C. T. Wang and C. C. Chen, “A combined optimization method for solving the inverse kinematics problems of mechanical manipulators,” *IEEE Trans. Robot. Autom.*, vol. 7, no. 4, pp. 489–499, Aug. 1991.

[12] J. B. Kuipers, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*. Princeton, N.J.: Princeton University Press, 2002.

[13] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, Édition: 3rd ed. 2008. Berlin: Springer-Verlag Berlin and Heidelberg GmbH & Co. K, 2008.

[14] J. O’Rourke, “Finding minimal enclosing boxes,” *Int. J. Comput. Inf. Sci.*, vol. 14, no. 3, pp. 183–199, Jun. 1985.

[15] M. Hariri, “A study of optimization-based predictive dynamics method for digital human modeling,” University of Iowa, 2012.

[16] S. Laine and T. Karras, “Efficient Sparse Voxel Octrees,” in *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, New York, NY, USA, 2010, pp. 55–63.