ロボットの頭の動作モデルの分析及びシミュレーション

コース　基盤科学コース
学籍番号　100052
氏名　磯崎陽一
指導教員　ルジェロ・ミケレット

Recently, many robots that work at factory instead of human are developed. In 20 years or more humans might coexist with humanoid robots. However, it is difficult to recognize machine as human being for now when a human look at humanoid robot. It finds out that it is very important to control actively head movement when human recognize human beings as same kind. So, in a collaboration with a robotics group of the university of Osaka, our goal is to find a new algorithm in order to active head movement naturally with mathematics on Python and Blender.

1．研究背景と目的

近年、人間型のロボットが社会福祉での活動や企業のブランドなどで使われるようになっている。今後、人間とロボットが同じ道で歩き、共に仕事を行い、共存するという未来が訪れるかもしれない。しかし現在の我々人間は人間型ロボットを見た時に、人間であると認識することはほとんどないだろう。それはロボットが人間のようなナチュラルな動きをしていないからである。

人間が人間を認識する際に、頭部の動きが認識においてとても重要であることがわかっている。そこで本研究では大阪大学の知能ロボット学研究室と共同研究を行い、実際のロボットが人間の頭部のように自然に動かせるような仕組みを開発することが目的である。

2．実験方法

図1のようにプログラミング言語であるPythonを用いて3D制作ソフトであるBlenderでの動作シミュレーションを行う。

図1　シミュレーションモデル展開図

本研究では頭部動作の初期位置と到着位置は自分で決め、さまざまな動き方をするモデル9パターンを作成し、シミュレーションおよび検証を行なった。

3．実験結果

自ら作成したモデルの中から一番自然な動きをするモデルを選択し、ほかの1つのモデルと比較を行い、考察するため図2にグラフを添付した。自然な動きをするモデルが右側であり、ある1つのモデルの比較対象が左側である。またX軸、Y軸は行列成分の値を示し、Z軸が時間を表す。

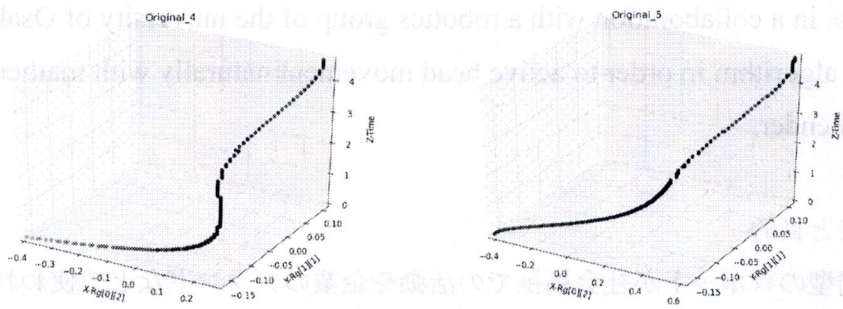

図2　比較対象のモデル（左）と自然な動きをするモデル（右）の散布図

4．結論

以上の結果より2つのことが言える。1つ目は散布図から見てわかるように、より直線に近いモデルの動きが自然である。2つ目は行列の成分を作成する際、三角関数を用いたモデルの方がシミュレーションを行なった時、実際の目と散布図のグラフから自然であると判断することが出来た。

しかし課題として、本研究では予め初期位置と到着位置を決め、その動作の過程を検証したが、今後ロボットが予め決めたれた方向へ向くのではなく、自らが向きたい方向に向く場合、まだ応用できないといった課題がある。しかし、このような数式を用いての動作を実際のロボットで動かし、検証していくことで、ロボットの頭部の動き方に新しいアルゴリズムとして認知される可能性がある。

<卒業論文>

「ロボットの頭の動作モデルの分析及びシミュレーション」

横浜市立大学　国際総合科学部　国際総合科学科　基盤科学コース

ルジェロ・ミケレット　研究室

磯崎　陽一

目次

第1章　序論

1.1　はじめに

近年、トヨタや日産など日本の自動車産業の自動走行車や工場などで稼働する産業ロボットが多く開発されている。自動走行車やロボットのメリットは人件費の削減や作業の効率化だけでなく、人間には難しい細かい作業や複雑な作業も難なくこなすことも可能になった。

また産業ロボットと違い人型のロボットの開発も昔に比べ多くなっている。それらは高齢者を対象とした介護ロボットやホンダ自動車でメインキャラクターとなっている ASIMO を含めブランド認知のためや癒しといった理由で開発されることも多くなった。

そうした背景のなか、20 年後もしくはもっとそれ以降のことになるかもしれないが、ロボットが街や道路で何かの作業を行なっている、また普通に道を歩いているといった人間とロボットが共存する未来が訪れるかもしれない。

2.2　研究背景と目的

人間が人型ロボットを認識するとき、あるいは人型ロボットが人間を認識するとき、どんな情報でこの物体は人間である、ロボットであると認識するのだろうか。人が人を認識するとき、服装や歩き方、眼球の動きなど様々な要素があるが、頭部の動きが認識においてとても重要であるとわかっている。

そのため、実際のロボットが人間の頭部のように自然に動かせるような仕組みを開発することを本研究の目的とする。ただし、実際にロボットを用いるのではなく、コンピュータ上でモデルを作成し、頭部のみの動きに焦点を合わせ、動作のシミュレーションを行なった。なお本研究において動作のモデルの作成及びシミュレーションを行う際に、プログラミング言語である Python と 3D 制作ソフトの Blender を用いて研究を行い、大阪大学に属する石黒教授の知能ロボット学研究室と共に共同で研究を行なった。

第 2 章　Blender ソフトウェアと Python 言語

2.1　Blender について

Blender とは 3D アニメ制作ソフトウェアであり、無料でインターネットからダウンロードすることが出来る。Blender は本来プログラミングを行わずに直感的に操作し、アニメーションを作ることが可能であるが、プログラミング言語である Python に対応しており、直感的な操作ではなくアルゴリズムを作成することで操作出来る一面を持っている。本研究はコードを書くことで操作する Python を用いて操作を行なった。

図 1 は実際の Blender の画面である。左の画面では実際のシミュレーションを確認できる画面であり、右側が Python 言語によるコード画面を表している。下に画面ではシミュレーションで用いる物体について詳細に設定を行える画面である。

なお実際にシミュレーションを行うにあたって、3×3 のダイナミック行列を用いてモデルを作成していくため、それら行列の変動値を得るため、図 2 のようなターミナルの画面を開き、エラー確認も合わせて研究準備を進めた。

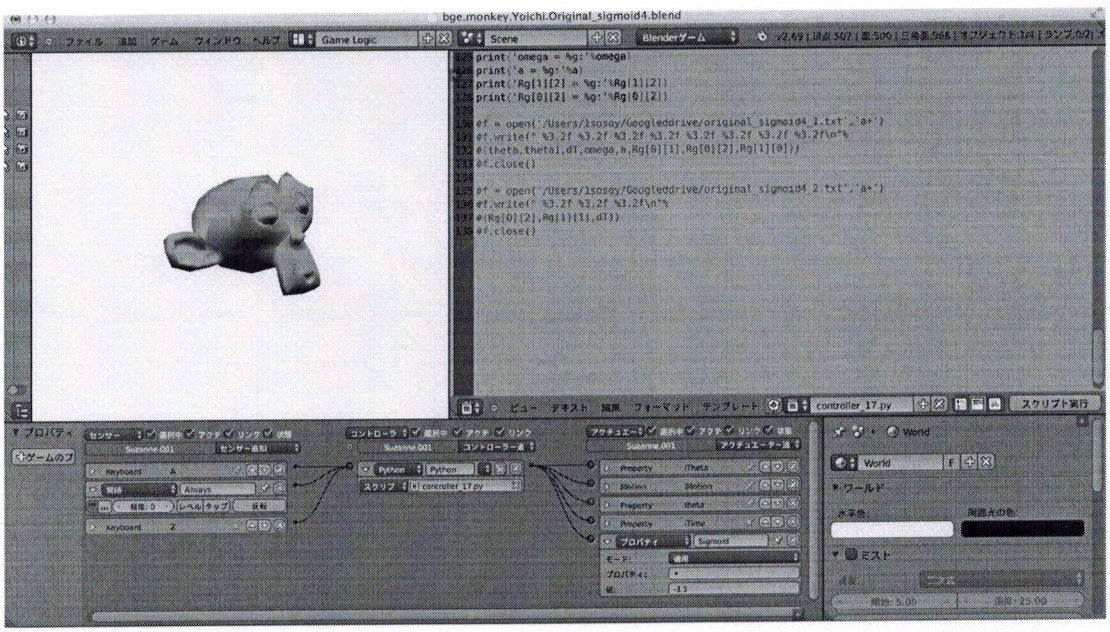

図 1 Blender 操作画面

```
omega = 0.01:
a = 1:
<Matrix 3x3 (0.7217, -0.1741,  0.1442)
            (0.1540, -0.0144, -0.7002)
            (0.0053,  0.9675,  0.0294)>
theta = 0.992047:
theta1 = 0.452642:
time = 8.0097:
omega = 0.01:
a = 1:
<Matrix 3x3 (0.7207, -0.1744,  0.1473)
            (0.1572, -0.0135, -0.6992)
            (0.0052,  0.9674,  0.0307)>
theta = 0.992127:
theta1 = 0.455121:
time = 8.02572:
omega = 0.01:
a = 1:
<Matrix 3x3 (0.7197, -0.1748,  0.1504)
            (0.1604, -0.0127, -0.6982)
            (0.0051,  0.9673,  0.0319)>
theta = 0.992205:
theta1 = 0.457602:
time = 8.04219:
```

図２　ダイナミック行列の値の変動確認画面（ターミナル）

2.2　概念

2.2.1　基本概念

本研究の目的は実際のロボットが人間のように自然に頭部を動かせるようにプログラミング言語を用いてシミュレーションを行うことである。実際の人間の頭部の動きではある方向からある方向を向くまでに様々な向き方が存在する（図３）。

よって予め、頭部のモデルである行列変数を R とし、初期位置を R_{START}、到着位置を R_{END} と定義する。

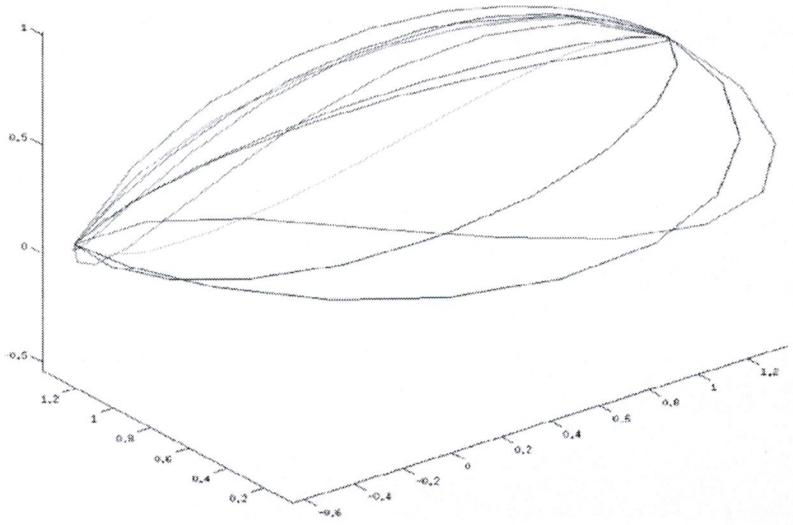

図３　様々な顔の向きの理論

頭部の動きを操作するために3×3のダイナミック行列を用いてシミュレーションを行なった。
その数式は

$$R_{END}\begin{pmatrix} X'_{11} & Y'_{12} & Z'_{13} \\ X'_{21} & Y'_{22} & Z'_{23} \\ X'_{31} & Y'_{32} & Z'_{33} \end{pmatrix} = R_X\begin{pmatrix} a_{11} & b_{12} & c_{13} \\ a_{21} & b_{22} & c_{23} \\ a_{31} & b_{32} & c_{33} \end{pmatrix} \times R_{START}\begin{pmatrix} X_{11} & Y_{12} & Z_{13} \\ X_{21} & Y_{22} & Z_{23} \\ X_{31} & Y_{32} & Y_{33} \end{pmatrix}$$

である。・・・(A)

R_{END} および R_{START} は予め自分で決めるため、R_X を様々な値に変えて研究を行う。

R_X の求め方は、

$$a_{11} = X_{11} + (X'_{11} - X_{11}) \times \quad \Theta$$
$$b_{12} = Y_{12} + (Y'_{12} - Y_{12}) \times \quad \Theta$$
$$_{13} = Z_{13} + (Z'_{13} - Z_{13}) \times \quad \Theta$$
$$\vdots$$

のように求めていく。・・・(B)

この θ は以降章の式を用いていくためここでは省略する。

2.2.2　大阪大学ファビオ准教授理論

序章でも触れたが、本研究は大阪大学と共同研究を行なった。実際に大阪大学を訪れ、大阪大学知能ロボット学研究室に属するファビオ教授から様々なアドバイスを受けた。ファビオ准教授はこのシミュレーションを行う上で

$$\theta = \theta + \omega(1 - \theta) \quad \cdots (C)$$

の式が適切であると可能性として提示していただいた。

ただし$(0 \leq \theta \leq 1)$

またωの値は変数であり、モデルの動作速度を表し、本研究では、

$$\omega = 0.1$$

として研究を行なった。

この式を視覚的に表すと図4のようになり、X軸は時間を表し、Y軸はθを表す。

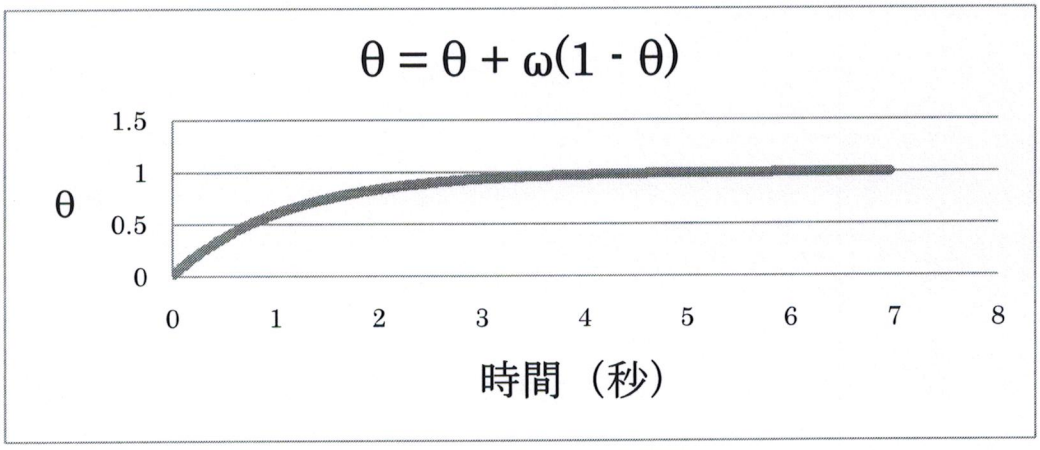

図4 ファビオ准教授モデル

2.2.3　シグモイド関数

シグモイド関数とは

$$\theta(x) = \frac{1}{1 - e^{-ax}} \quad \cdots \text{(D)}$$

であり、X の値が大きくなればなるほど 1 に近づき、X の値が小さくなればなるほど 0 になる。また X = 0 の時は $\frac{1}{2}$ になる。（図 5 参照）

なお本研究では a の値は

$$a = 1.0$$

として研究を行なった。

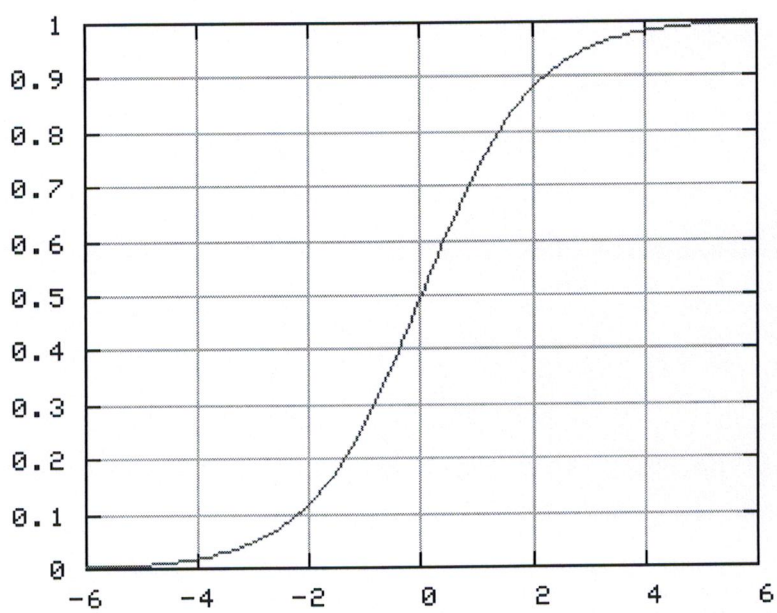

図 5　標準シグモイド関数

第３章　数式を用いてのシミュレーション

3.1　実験

3.1.1　実験準備

本実験では前述したようにPythonを用いてBlenderでのシミュレーションを行う。なお（A）で示した式のように予め初期位置であるR_{START}および到着位置R_{END}を以下のように決めた。

その値は

$$R_{START}\begin{pmatrix} 0.901 & -0.109 & -0.419 \\ -0.432 & -0.167 & -0.886 \\ 0.026 & 0.979 & -0.197 \end{pmatrix} \qquad R_{END}\begin{pmatrix} 0.504 & -0.252 & 0.825 \\ 0.863 & 0.170 & -0.475 \\ -0.020 & 0.952 & 0.304 \end{pmatrix}$$

である。

以下にシミュレーション時でのモデルの動きを図として示す（図6）

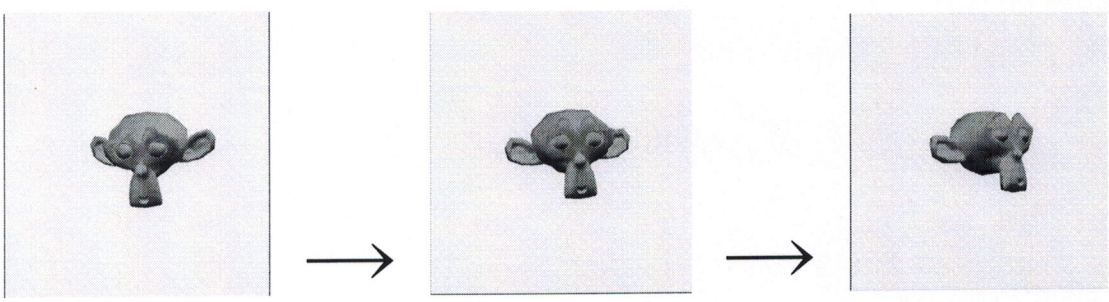

図6　シミュレーションモデル展開図

3.1.2　実験方法

本研究の実験では前述した式（A）を元に、（C）、（D）を組み合わせ（B）に代入し、モデル動作の自然さを追求した実験である。本実験を行うにあたって、全部で９つのモデルを作成した。それぞれのモデルを作成した後、ω、a、時間、そしてそれぞれの式の値を元に三次元のグラフを作成した。それらのモデルを元にどのモデルが自然体なのかを研究していく。

モデル①：それぞれの式（B）の θ に式（C）を代入する

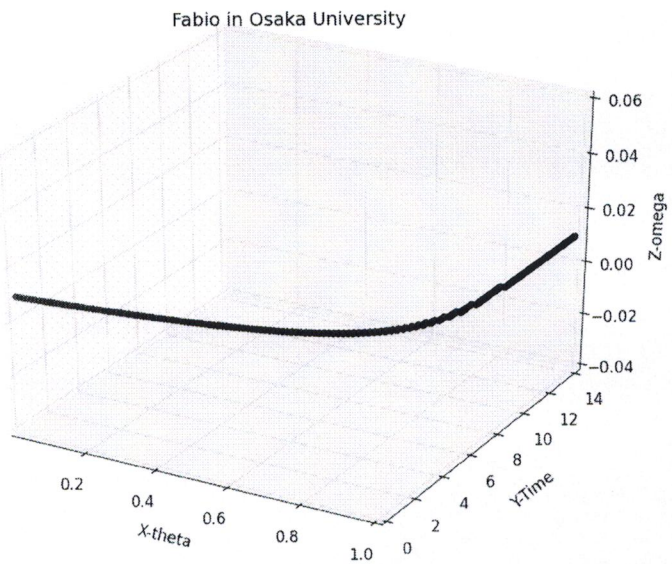

図７　モデル①：X 軸：式（C）、Y 軸：時間、Z 軸：ω

モデル②：それぞれの式（B）の θ に式（D）を代入する。

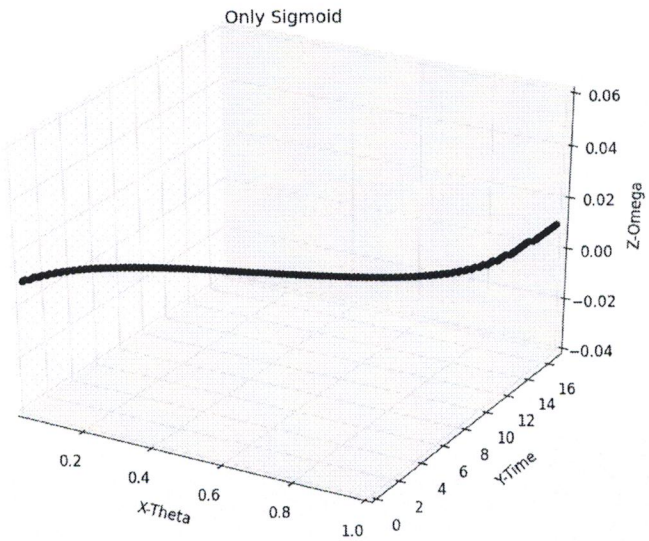

図 8　モデル②：　X 軸：式（D）、Y 軸：時間、Z 軸：ω

モデル③：$R_X\begin{pmatrix} a_{11} & b_{12} & c_{13} \\ a_{21} & b_{22} & c_{23} \\ a_{31} & b_{32} & c_{33} \end{pmatrix}$ での a_{11}、b_{22}、c_{33} の θ に式（D）を、それ以外の θ には式（C）を代入。

モデル③及び④はお互いを比較するため a_{11}、b_{22}、c_{33} の値をグラフにした。

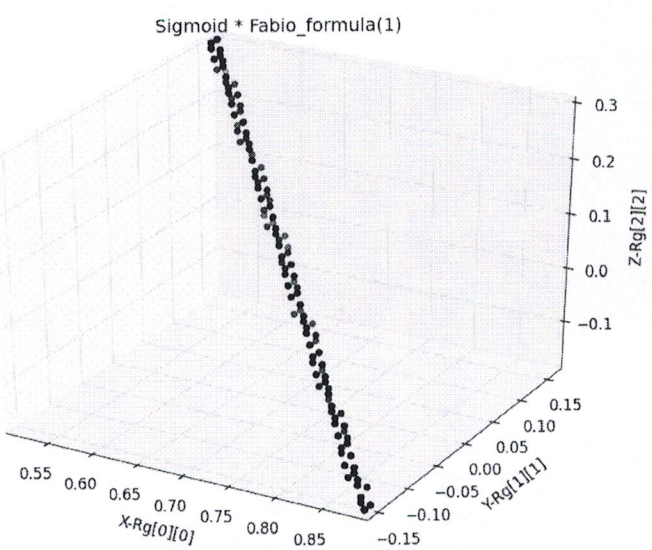

図 9　モデル③：X 軸：a_{11}、Y 軸：b_{22}、Z 軸：c_{33}

モデル④：$R_X\begin{pmatrix} a_{11} & b_{12} & c_{13} \\ a_{21} & b_{22} & c_{23} \\ a_{31} & b_{32} & c_{33} \end{pmatrix}$ での b_{22}、c_{23}、b_{32}、c_{33} の θ に式（D）を、それ以外の θ には式（C）を代入。

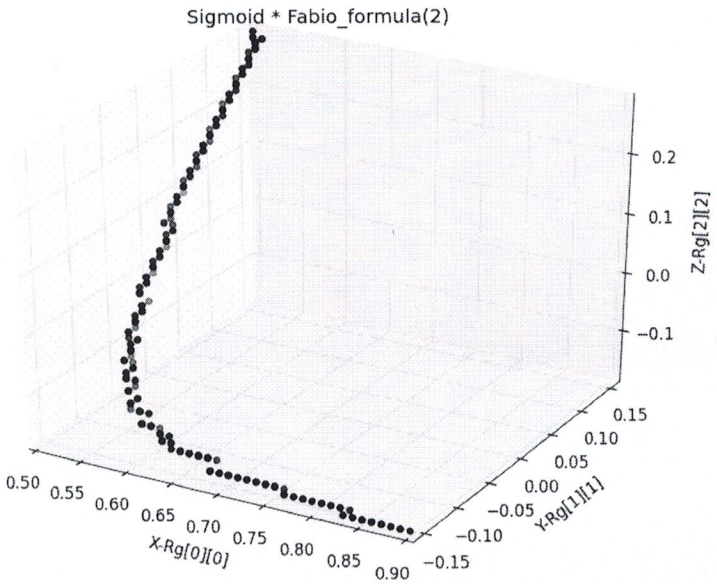

図 10　モデル④：　X 軸：a_{11}、Y 軸：b_{22}、Z 軸：c_{33}

図 11 ではモデル③とモデル④とを同じ行列成分で比較した。

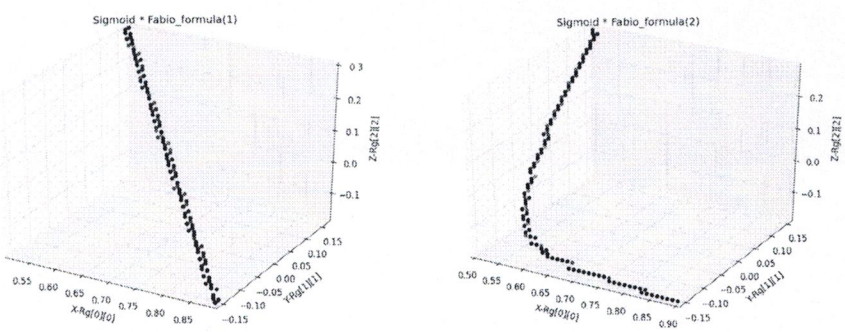

図 11　モデル③とモデル④の比較　X 軸：a_{11}、Y 軸：b_{22}、Z 軸：c_{33}

モデル⑤:モデル④とあまり変わらないが、$R_X\begin{pmatrix} a_{11} & b_{12} & c_{13} \\ a_{21} & b_{22} & c_{23} \\ a_{31} & b_{32} & c_{33} \end{pmatrix}$ での b_{12} に式(D)を、c_{23} に式（−D）を代入し、a_{11} の θ に式（−C）を、それ以外の θ には式（C）を代入する。

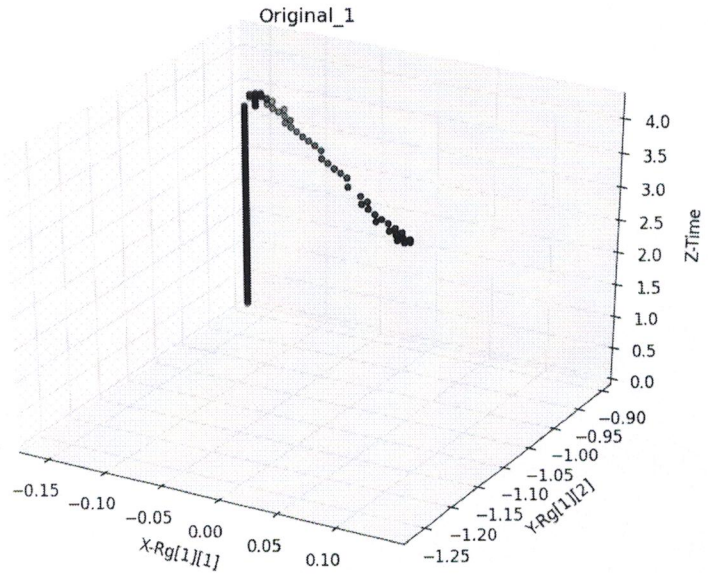

図 12　モデル⑤：　X 軸：b_{22}、Y 軸：c_{23}、Z 軸：時間

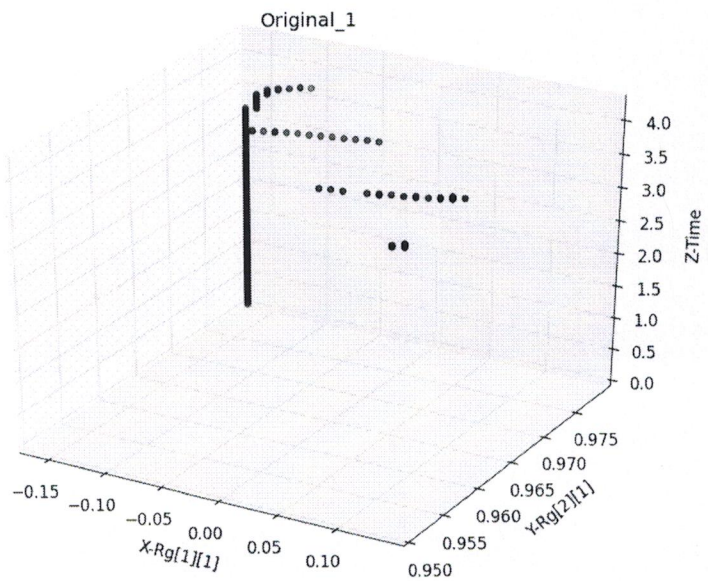

図 13　モデル⑤ X 軸：b_{22}、Y 軸：b_{32}、Z 軸：時間

モデル⑥：モデル⑥からは式（C）（D）に加え三角関数を追加して作成を行なった。

$R_X \begin{pmatrix} a_{11} & b_{12} & c_{13} \\ a_{21} & b_{22} & c_{23} \\ a_{31} & b_{32} & c_{33} \end{pmatrix}$ の b_{22} での θ には(D)×sin()を c_{23} での θ の値に(D)×-cos()を代入。b_{12} の θ には式（D）を代入、b_{32} の θ には式（C）×-cos()、c_{33} の θ には（C）×sin()を代入し、a_{11} の θ には式（−C）を代入する。

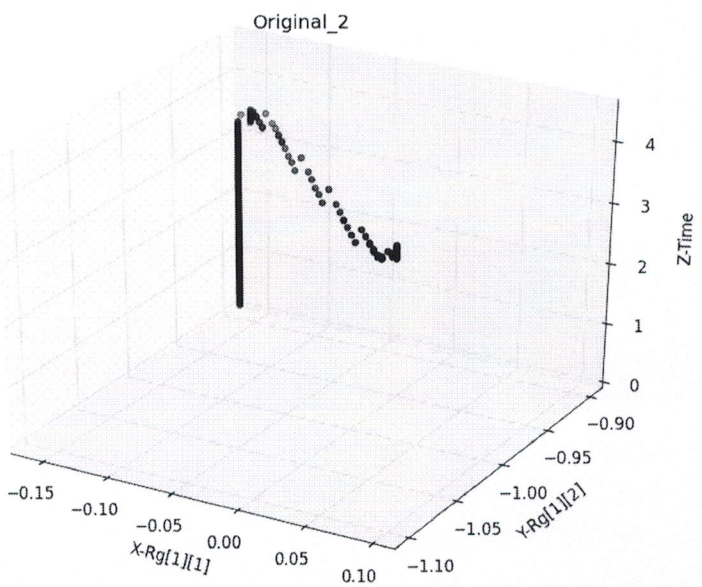

図 14　モデル⑥：X 軸：b_{22}、Y 軸：c_{23}、Z 軸：時間

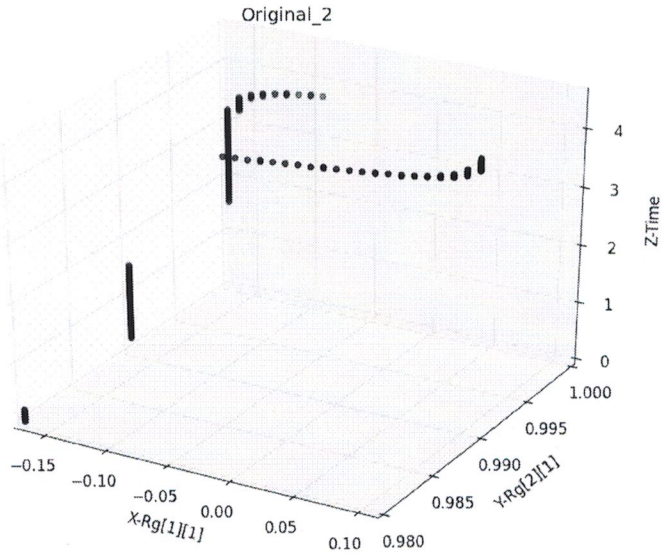

図 15　モデル⑥：X 軸：b_{22}、Y 軸：b_{32}、Z 軸：時間

図 16 ではモデル⑤とモデル⑥とを同じ行列成分で比較した。

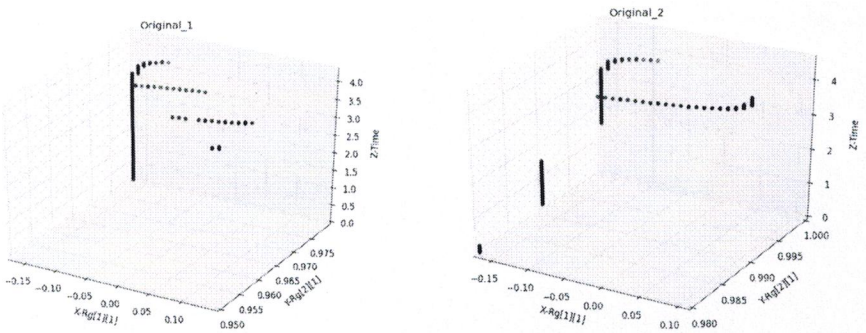

図 16　モデル⑤とモデル⑥の比較　X 軸：b_{22}、Y 軸：b_{32}、Z 軸：時間

モデル⑦：モデル⑥では c_{23} の θ には式（D）×(−cos())を代入していたが、モデル⑦では c_{23} の θ には式（D）×（-sin()）を代入する。それ以外はモデル⑥と同じである。

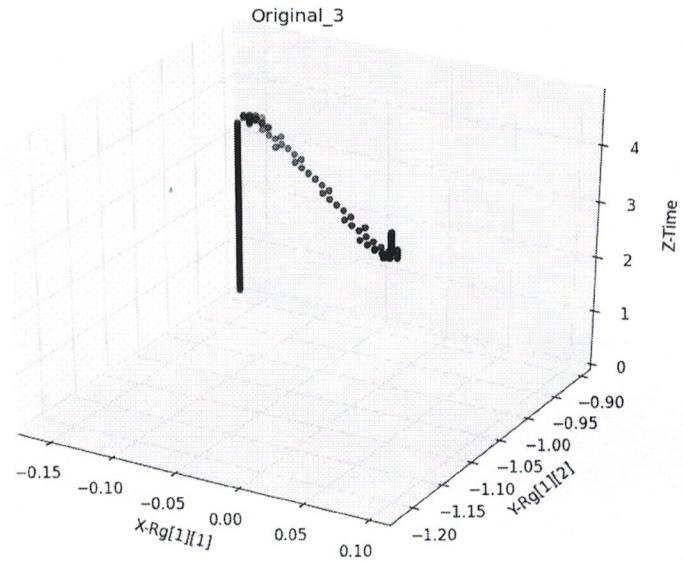

図 17　モデル⑦：　X 軸：b_{22}、Y 軸：c_{23}、Z 軸：時間

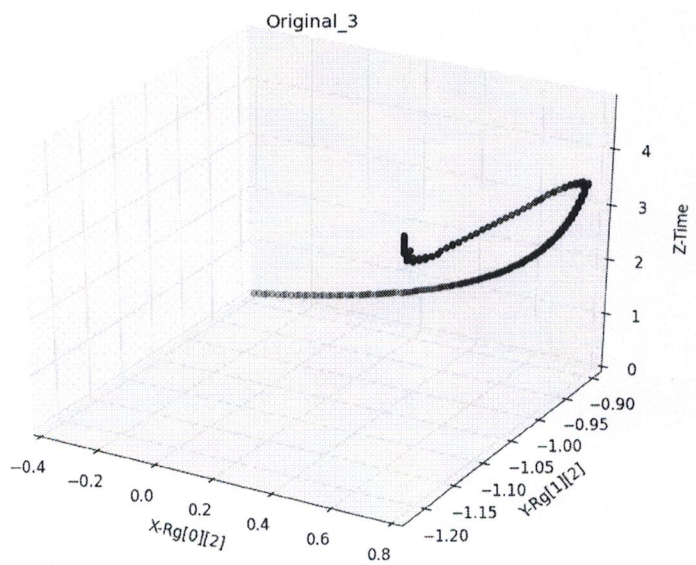

図 18　モデル⑦：X 軸：c_{13}、Y 軸：c_{23}、Z 軸：時間

図 19 ではモデル⑥とモデル⑦とを同じ行列成分で比較した。

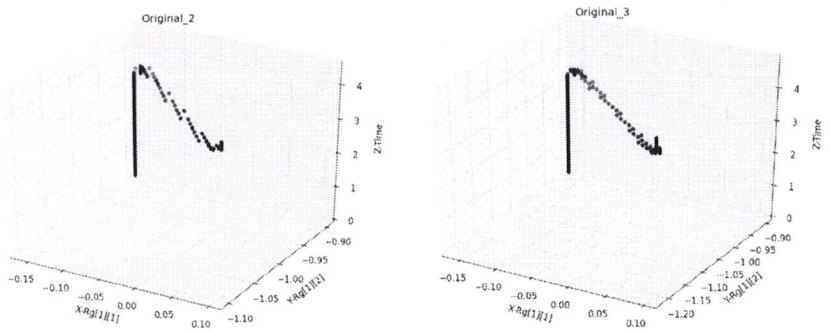

図 19　モデル⑥とモデル⑦の比較　X 軸：b_{22}、Y 軸：c_{23}、Z 軸：時間

モデル⑧：モデル⑦では c_{13} の θ に式（C）だけであったが、モデル⑧では c_{13} の θ に式（C）× cos()を代入する。それ以外はモデル⑦と同じである。

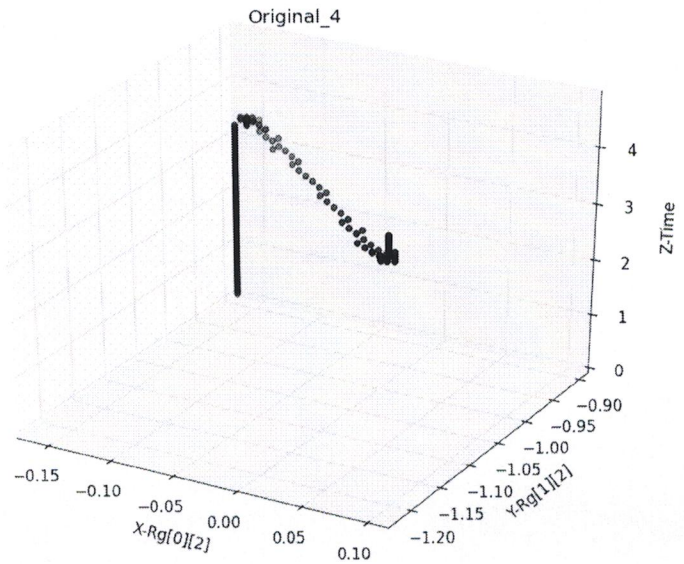

図 20　モデル⑧：X 軸：c_{13}、Y 軸：c_{23}、Z 軸：時間

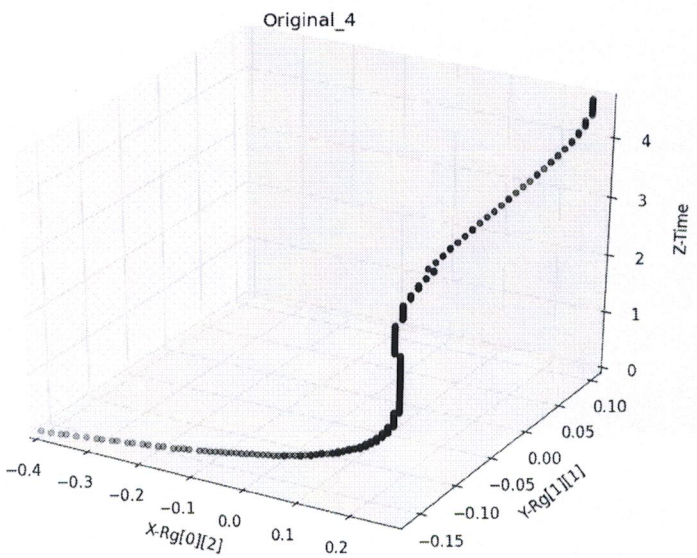

図 21　モデル⑧：X 軸：c_{13}、Y 軸：b_{22}、Z 軸：時間

図 22 ではモデル⑦とモデル⑧を同じ行列成分で比較した。

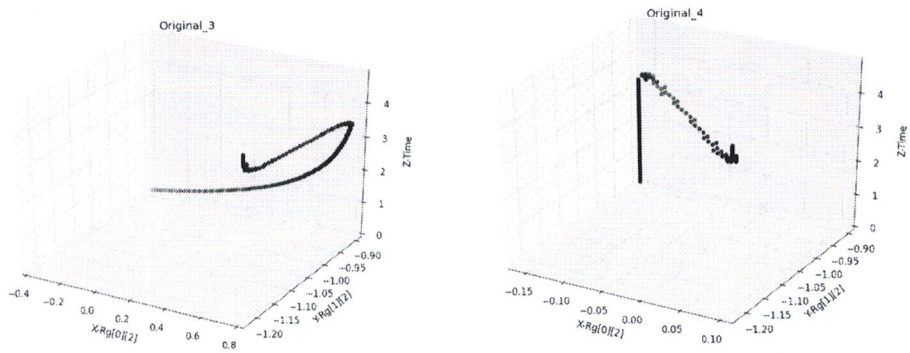

図 22　モデル⑦とモデル⑧の比較　X 軸：c_{13}、Y 軸：c_{23}、Z 軸：時間

モデル⑨：$R_X\begin{pmatrix} a_{11} & b_{12} & c_{13} \\ a_{21} & b_{22} & c_{23} \\ a_{31} & b_{32} & c_{33} \end{pmatrix}$ での b_{12} の θ には式（C）を、c_{13} と a_{21} の θ には式（C）×sin()を代入し、そのほかはモデル⑧と同じである。

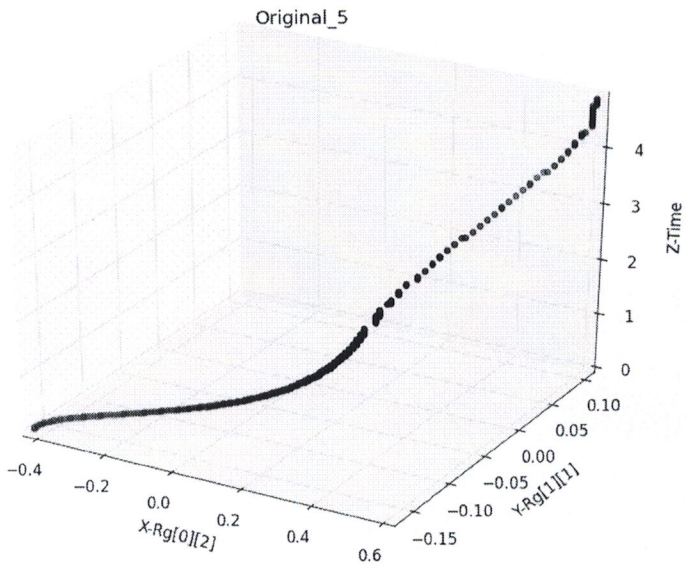

図 23　モデル⑨：X 軸：c_{13}、Y 軸：b_{22}、Z 軸：時間

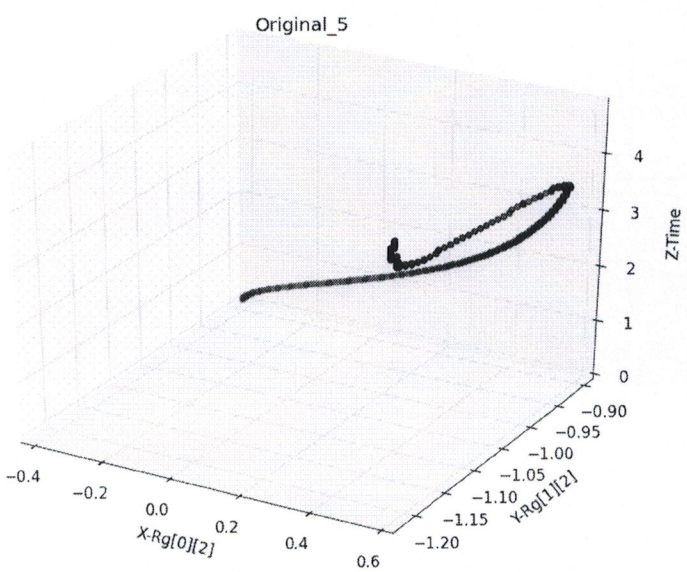

図 24　モデル⑨：X 軸：c_{13}、Y 軸：c_{23}、Z 軸：時間

図 25 ではモデル⑧とモデル⑨とを同じ行列成分で比較した。

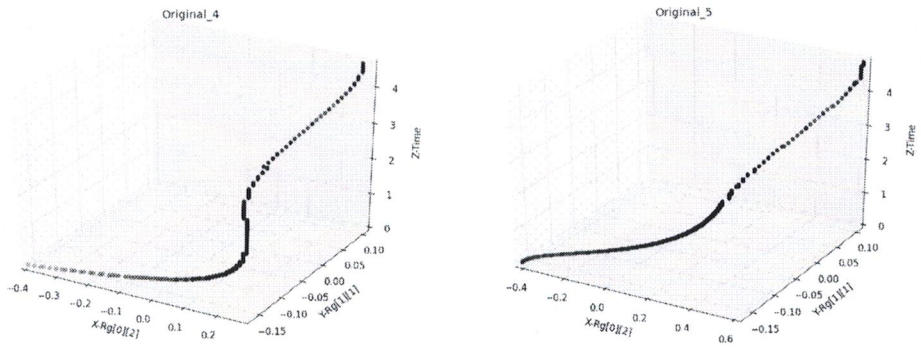

図 25　モデル⑧とモデル⑨の比較　X 軸：c_{13}、Y 軸：c_{23}、Z 軸：時間

3.1.3　実験結果

上記の実験方法でそれぞれのモデルを作成した結果自分の主観的な目線でどのモデルが一番人間らしさのあるモデルなのか比較した結果、モデル⑨が一番人間らしさというものが感じられたという結果になった。

モデル①から⑨まで作成を行なったが、③と④を除き、モデルの番号が高くなるにつれてナチュラルさが表れたのではないかと判断をしている。そうした結果、本研究では、ファビオ准教授からアドバイスいただいた式（C）およびシグモイド関数である式（D）のみを使用したが、それに加え三角関数を用いて変化を加えたモデルの方がよりナチュラルな動きになった。

3.1.4　考察

実験結果より、モデル⑨が一番人間らしさのある動きという結果になったが、そう判断できる理由として、グラフからも２点読み取ることが出来る。１点目は複雑な散布図になっているグラフよりも直線に近いグラフのほうがより自然な動きをするということだ。図 11 を見るとわかるようにシミュレーションをしてみてモデル④よりもモデル③の方が自然な動き方をする。

２点目として図 16 では三角関数を用いたモデル⑤と三角関数を用いたモデル⑥を比較したが、三角関数を用いたモデル⑥の方がシミュレーションでもより自然であると判断でき、それはグラフからも視覚的に判断することが出来る。

またモデル⑧では$R_X\begin{pmatrix} a_{11} & b_{12} & c_{13} \\ a_{21} & b_{22} & c_{23} \\ a_{31} & b_{32} & c_{33} \end{pmatrix}$の c_{13} に cos()を代入したが、モデル⑨では sin()を代入して比較を行なったわけだが、図 25 でもわかるように c_{13} に sin()を代入した方がグラフから視覚的に、シミュレーションの動作でも判断することが出来た。

以上のことから、より人間らしさを追求するには、今回は式（C）（D）しか用いなかったが、グラフでの直線さを求め、また三角関数を用いる必要があると言える。

第４章　結論

本研究ではプログラミング言語である Python を用いて、３D 制作ソフトである Blender の中でいくつかのモデルを作成し、実際に動作の確認でシミュレーションを行い、人間らしさを追求する研究を示した。結論として、実際に Python と Blender を用いて人間の頭部の動きを作成することに成功し、より人間らしさを追求した動きを作成することが出来た。

課題としては、３点挙げられる。１点目として本研究では大阪大学の知能ロボット学研究室に在学するファビオ准教授からアドバイスを受けた上記、式（C）およびにシグモイド関数である式（D）のみしか使用していないこと。もっと人間らしい動きをする式がある可能性が十分にあると考えられる

２点目では本研究は初期位置と到着位置を自分で決め、その過程を研究することに焦点が当てられたが、到着位置を決めずに、好きな方向へ人間らしく動かすことを行なっていないので、ロボットが意思も持ち頭部を動かす場合、まだ応用が出来ない。

３点目はグラフを実際に作成し、視覚的に判断することが出来たが、自分が作成した９つのモデルをもっと改良を行えば、今以上のモデルを作成することが出来た可能性もある。

今後の展望として、コンピュータ上でのシミュレーションではなく実際のロボットのモデルを扱い、研究を行うことを検討していきたい。

謝辞

本研究を行うにあたって、豊富な知識とアイデアをお持ちのルジェロ・ミケレット教授には終始丁寧なご指導をいただき感謝いたします。そして、お忙しい中、私の研究のため親身に協力していただいた大阪大学知能ロボット学研究室のファビオ准教授に厚く御礼申し上げます。また些細な研究トラブルに丁寧に相談に乗っていただいた同研究室の孫哲さんに心から感謝致します。

最後に親身になって支えていただいた教授、先輩方、同期がいる環境で研究を行えたことに、感謝の意を申し上げて、本論文の結びとさせて頂きます。

参考文献

[1] 「Python スタートブック」
技術評論社 / 著 辻真吾

[2] 「みんなの Python 第３版」
SoftBank Creative / 著 柴田淳

[3] 「無料でできる３D アニメーション ブレンダーからはじめよう！」
技術評論社 / 著 原田大輔

[4] 「Optimal Control of Natural Eye-Head Movements Minimizes the Impact of Noise」
Behavioral/Systems/Cognitive / 著 Murat Salram, Nadine Lehnen , and Stefan Glasauer

参考付録

```
/*************************************************
/モデル⑨を Blender で作成する Python プログラム
/作成者　磯崎 陽一、ルジェロ・ミケレット教授
/*************************************************

import bge
import time
from math import *
import bpy


cont = bge.logic.getCurrentController()

//物理時間を得るコマンド
iT = cont.actuators['iTime']
dT = time.time() -float(iT.value)
if (dT>100):
    t0=time.time()
    iT.value=str(t0)

//図 1 でのそれぞれのオブジェクトの詳細を
    作ることができるコマンド
own=cont.owner
it=cont.actuators["theta"]
X = cont.actuators['Sigmoid']
```

```
theta=float(eval(it.value))
x = float(eval(X.value))

//初期位置
oStart=[[0.9013,-0.109,-0.4193],
        [-0.4325,-0.167,-0.8861],
        [0.0266,0.9799,-0.1976]]

//到着位置
oEnd=[[0.5045,-0.2528,0.8256],
      [0.8632,0.1702,-0.4754],
      [-0.0204,0.9524,0.3040]]


ioStart=inv(oStart)
Rx=mult(oEnd,ioStart)

//式（C）の作成コマンド
omega = 0.1
a = 1.0
theta = theta + omega*(1-theta)

//式（D）の作成コマンド
x = x + omega
X.value = str(x) # store in memory
theta1 = 1/(1 + exp(-a * x))

//sin(),cos()作成コマンド
sce = bpy.context.scene
sce['theta'] = theta
s = sin(theta)
c = cos(theta)
it.value=str(theta)
```

```
Rg = [[0,0,0], [0,0,0], [0,0,0]]

//行列式 Rx のそれぞれの成分
Rg[0][0]=oStart[0][0]+(oEnd[0][0]-oStart[0][0])*-theta
Rg[1][0]=oStart[1][0]+(oEnd[1][0]-oStart[1][0])*theta*s
Rg[2][0]=oStart[2][0]+(oEnd[2][0]-oStart[2][0])*theta
Rg[0][1]=oStart[0][1]+(oEnd[0][1]-oStart[0][1])*theta
Rg[1][1]=oStart[1][1]+(oEnd[1][1]-oStart[1][1])*theta1*s
Rg[2][1]=oStart[2][1]+(oEnd[2][1]-oStart[2][1])*-theta*c
Rg[0][2]=oStart[0][2]+(oEnd[0][2]-oStart[0][2])*theta*s
Rg[1][2]=oStart[1][2]+(oEnd[1][2]-oStart[1][2])*-theta1*s
Rg[2][2]=oStart[2][2]+(oEnd[2][2]-oStart[2][2])*theta*s
own.worldOrientation = Rg

//それぞれの値をターミナルに表示
print(own.worldOrientation)
print('theta = %g:'%theta)
print('theta1 = %g:'%theta1)
print("time = %g:"%dT)
print('omega = %g:'%omega)
print('a = %g:'%a)

//値をテキストファイルにするコマンド
f = open('/Users/1sosoy/Googleddrive/original_sigmoid5_2.txt','a+')
f.write(" %3.2f %3.2f %3.2f¥n"%
(Rg[0][2],Rg[1][1],dT))
f.close()
```

```
/*************************************************
/テキストファイルから三次元グラフにする Python プログラム
/作成者　磯崎 陽一
/*************************************************
from numpy import *
from matplotlib import pyplot
from mpl_toolkits.mplot3d import Axes3D
from scipy import genfromtxt

fig = pyplot.figure()
ax = Axes3D(fig)

//テキストファイル読み込み
data = loadtxt('original_sigmoid5_2.txt')

//変数に代入
Rg_02 = data[:,0]
Rg_11 = data[:,1]
time = data[:,2]

//タイトルおよび軸の名前
ax.set_xlabel("X-Rg[0][2]")
ax.set_ylabel("Y-Rg[1][1]")
ax.set_zlabel("Z-Time")
ax.set_title('Original_5')

//実行
ax.scatter3D(ravel(Rg_02),ravel(Rg_11),ravel(time))
pyplot.show()
```