Acta Univ. Sapientiae, Informatica **11,** 1 (2019) 80–94

DOI: 10.2478/ausi-2019-0006

Gesture-Driven LEGO robots

Lehel István KOVÁCS
Sapientia Hungarian University of Transylvania, Cluj-Napoca
Department of Mathematics and Informatics,
Târgu-Mureş, Romania
email: klehel@ms.sapientia.ro

Abstract. In this short survey and case study we want to present our research experience through the project developed by our team, that involves the building of a LEGO MINDSTORMS EV3 robotic arm and tracked robot car which mimics the motion of the human arm and legs. We used 3 interconnected LEGO MINDSTORMS EV3 bricks to reach the desired degrees of freedom. Using a Kinect sensor, the system detects the motion of the human user's arm and creates the skeletal image of the arm. Coordinate geometry and different approximation methods are used to calculate the rotation angles between the bones connecting the joints. In our project the key is inverse kinematics, whitch makes use of the kinematics equations to determine the joint rotation parameters that provide a desired position for each of the robot's end-effectors – arms and legs (wheels). The combined motion of the LEGO MINDSTORMS EV3 motors results in a complete robotic forward or backward motion and arm movement which is a perfect mimic of the human arm movement.

1 Introduction and motivation

The purpose of this paper is to introduce and compare different implementations of inverse kinematics, and to present a LEGO [16] robot and Kinect

Computing Classification System 1998: J.1.3
Mathematics Subject Classification 2010: 70E60
Key words and phrases: human-robot interaction, robotics, end-effectors, Kinect, LEGO MINDSTORMS EV3.

[15] (for motion capture) system that mimics the motion of the human arm as a case study. Our application implements five different algorithms to solve inverse kinematics problem. We compare and analyze the algorithms to select the most appropriate one.

Inverse kinematics as an animation technique plays an important role in computer animation and as we will see, robots can be controlled with the help of it.

During the last decades several algorithms have been presented to produce fast and realistic solutions to the inverse kinematics problem, but most of the available algorithms have high computational cost and produce unrealistic poses. The [1] technical report summarizes the research to date and the results achieved and methods developed. Based on the [10] book, using OpenGL [17], we have developed a graphical system that displays the results of the movements, the animation.

Kinect is used to copy human movements, we recognize gestures, and execute them with our specially built LEGO robot.

Most of the results achieved have been presented at conferences [11], and details and new solutions will be presented.

The body of this survey paper is divided into 6 sections. The first section introduce readers to the robot control and animation, the second section describes the rotation in plane and space as the basis of animation. Section 3 presents the Inverse Kinematics (IK) problem and discusses the most popular and developed solutions during the last couple of decades. Section 4 presents the Kinect sensor and introduces gesture recognition methods. And finally, Section 5 describes how our built LEGO robot works. The final section summarizes the conclusions of our work.

2 Robot control and animation

Robot control is the study and practice of controlling robots [3]. Animation is a method in which standing pictures are manipulated by quick changes to appear as moving images [13]. Commonly the effect of animation (the movie, or motion) is achieved by a rapid succession of sequential images – drawn or computer generated – that minimally differ from each other.

But what does robot control and animation have in common?

If we want to answer the question briefly, we could say that the techniques. In the longer term, we need to look specifically at the techniques and how to apply them.

When we talk about animation techniques, we distinguish between simple and complex animation. *Simple animation* is the *key frame animation* or the *program-controlled animation*. A keyframe in computer assisted or aided animation and cinematography is a drawing suite that defines the starting, the middle and ending points of any smooth transition [19]. The drawings, the pictures (the given particular images) are called "frames" because their position in time is measured in frames on a strip of old type of cellulose film.

In program controlled or program-driven animation, we write scripts that implement the animation. *Complex animation* is *motion caption*, *forward kinematics* and *inverse kinematics*. Motion capture (mo-cap or mocap) is the process of recording the movement of objects or people using differend kind of sensors. In animation and film making, it refers to recording actions (movements) of human or in some cases, not human actors, and using that information to animate digital character models (skeletons) in 2D or 3D computer aided animation. If we use Kinect or VR equipment, we can realize a full robot control using motion capture. The disadvantage of the method is, that only human-shaped (android, humanoid) robots can be controlled in this way. But how to control for example a spider- or dog-shaped robot? For these cases, and not only, forward kinematics and inverse kinematics are suitable. For these animations we need a bone/joint system.

In computer aided animation a designed figure, character is represented in two parts: a surface representation (called *skin* or *mesh*), this is used to draw the character and a hierarchical, recursive set of interconnected bones (called the *skeleton* or *rig*) used to animate the figure, the character.

Each bone has a 2D (in plane) or 3D (in space) transformation (which includes its position, scale and orientation in plane, space), and an optional parent bone. The bones therefore form a hierarchy. This is a hierarchical and recursive structure, because the full transform of a child node is the product of its parent transform and in plus its own transform. So moving a thigh-bone will move the lower leg too. As the character is animated during the process, the bones change automatically their transformation over time, under the influence of some animation controller [12].

The essential concept of forward kinematic animation – one of the animation techniques – is that the positions of particular parts of the model at a specified time are calculated from the position and orientation of the object, together with any information on the joints of an articulated model. In this model the animator must to calculate each position of every joints. So for example if the object to be animated is an arm with the shoulder remaining at a fixed location, the location of the tip of the thumb would be calculated

from the angles of the shoulder, elbow, wrist, thumb and knuckle joints. The avantage of forward kinematic is, that we have a very precise control [18]. The disadvantage: difficult for complex movements, for example we can not climbing stairs with forward kinematic control.

For complex movements the effective method ist the inverse kinematics. In kinematics an animated figure is modeled with a skeleton of rigid segments connected with joints, called a *kinematic chain*. The kinematics equations of the figure define the relationship between the joint angles of the figure and its pose or configuration. In inverse kinematics the orientation of articulated parts is calculated from the desired position of certain points on the model.We move only the end effector, and the program calculates the kinematic path of each bones. The joints bind the bones. The bone rotates around the joint in a given range of angle. The length of the bones does not change during the movement. Mathematically, the system is complex. The exact method exists for only two bones, the system becomes very complicated for more bones. Nonlinear optimization is required.

In our project we use inverse kinematics for robot controll, but also motion caption is required.

3 Rotation in plane (2D) and space (3D)

As we have seen, motion is accomplished by inverse kinematics, which is obtained with rotation of bones around the joints. Rotations are described using *Euler angles*. The Euler angles are three angles introduced by Leonhard Euler to describe the orientation in space of a rigid body (described by the *pitch*, *yaw* and *roll* movements) with respect to a fixed coordinate system. Any orientation can be achieved by composing three elemental rotations. We use the following angles: ψ around Z, θ around X, and φ around Y axis. In the plane, rotation is accomplished by complex numbers. The multiplication rules for complex numbers make them suitable for representing rotational quantities in two dimensions. Using complex numbers, we can describe rotations as follows: $CR = C_1 \cdot C_2$, where C_1 is a complex number representing the initial orientation, C_2 is a complex number representing the subsequent rotation, and CR is a complex number representing the final orientation. If we want to rotate with two angles, first one with θ, after that with φ, we can use the exponential form of the complex number: $z = r \cdot e^{i\theta}$. Then the rotation can be traced back to the product of two complex numbers because: $e^{i(\theta+\varphi)} = e^{i\theta} \cdot e^{i\varphi}$. If we

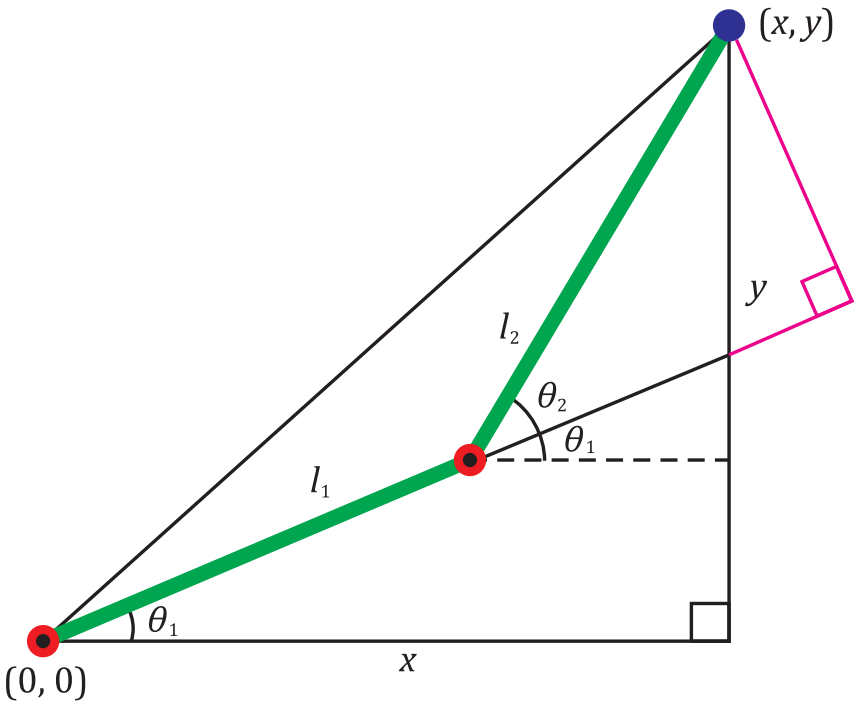

Figure 1: The analytical solution

want to repeat the rotation, the De Moivre theorem will simplify the situation: $(\cos\theta + i \cdot \sin\theta)^n = \cos n\theta + i \sin n\theta$.

In the space (3D) instead of complex numbers, we use *quaternions*. In mathematics, the quaternions are a number system that extends the complex numbers. They were first described by Irish mathematician William Rowan Hamilton in 1843 and applied to mechanics in three-dimensional space. Hamilton defined a quaternion as the quotient of two directed lines in a three-dimensional space or equivalently as the quotient of two vectors. The general form of a quaternion: $q = w + xi + yj + zk$, where $i^2 = j^2 = k^2 = -1$, and $ji = -k$, $kj = -i$, $ik = -j$. Thus, all operations can be defined on quaternions. Quaternions can be used to explain orientation and quaternion operations to track its changes.

For a given rotation axis, the rotation formula, using Euler angels, is:

$$\begin{aligned} q &= \cos\frac{\varphi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \sin\frac{\varphi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} \\ &\quad + i\cdot\left(\cos\frac{\varphi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\varphi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2}\right) \\ &\quad + j\cdot\left(-\cos\frac{\varphi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} + \sin\frac{\varphi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2}\right) \\ &\quad + k\cdot\left(\sin\frac{\varphi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \cos\frac{\varphi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2}\right). \end{aligned}$$

4 The solutions of inverse kinematic problem

As we said, the exact method to solve the inverse kinematic problem exists only for two bones, the system becomes very complicated for more bones. This is the *analytical solution*.

Problem statement:

There are two bones given, the first has one end in the origin $(0,0)$ and the second bone starts at the other end. With what kind of angles do you need to rotate the two bones so that the free end of the second bone reaches a given position (x,y)?

The solution of the problem, according to Figure 1 is:

$$\cos\theta_2 = \frac{x^2+y^2-l_1^2-l_2^2}{2l_1l_2}$$

and

$$\tan\theta_1 = \frac{y\cdot(l_1+l_2\cdot\cos\theta) - x\cdot l_2\cdot\sin\theta_2}{x\cdot(l_1+l_2\cdot\cos\theta) + y\cdot l_2\cdot\sin\theta_2}$$

Note: instead of $\mathtt{atan}$ function in computer science we use the $\mathtt{atan2}$ function. The definition of this function is:

$$\mathtt{atan2}(y,x) = \begin{cases} \mathtt{atan}(y/x), \text{ if } x>0, \\ \mathtt{atan}(y/x)+\pi, \text{ if } x<0 \text{ and } y\geq 0, \\ \mathtt{atan}(y/x)-\pi, \text{ if } x<0 \text{ and } y<0, \\ +\frac{\pi}{2}, \text{ if } x=0 \text{ and } y>0, \\ -\frac{\pi}{2}, \text{ if } x=0 \text{ and } y<0, \\ \text{undefined, if } x=0 \text{ and } y=0. \end{cases}$$

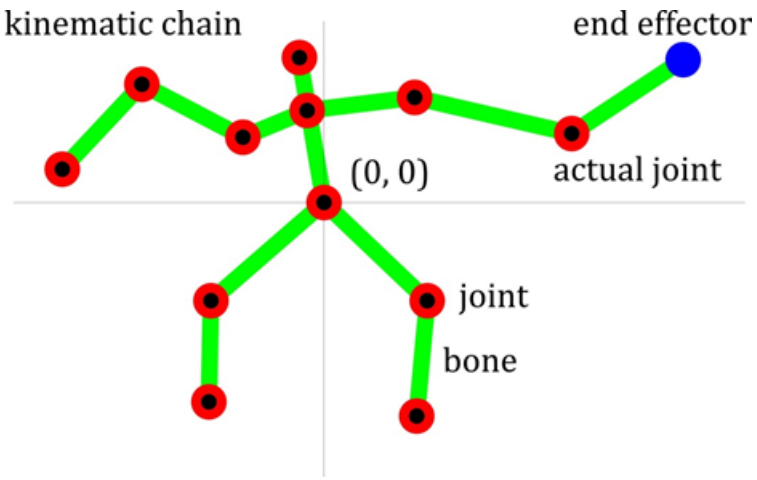

Figure 2: Kinematic chain – hierarchical movement

For more than two bones, we can solve the system by hierarchical movement based on Figure 2. We are iterating from the end effector to the base and optimizing each joint so that the last one comes close as possible to the target. So, we can have the same solution for more inverse kinematics problem.

Problem statement:

Known: The coordinates of end effector: $e = [e_1 e_2 ... e_N]$.

Unknown: The degree of freedom (DOF): $\theta = [\theta_1 \theta_2 ... \theta_M]$.

The solution of the problem: $\theta = f^{-1}(e)$.

Problems encountered: The f function is nonlinear. Calculation of the inverse function is not trivial. The solution is ambiguous, multiple states may have the same effector position.

The nonlinearity and ambiguity of inversion can be solved by an iteration that produces one of the possible solutions. The base idea of the iteration: if we known the position of end effector in a time moment t, we can calculate the position in time moment $t + \Delta t$. If Δt is small, we can approximate the function with its tangent (linear approximation).

In our research, a comparative analysis of four methods was made in this sense. We compared the approximation method based on *Jacobian matrix* [4], the correction of this method called *Damped Least Squares* [4], the *Cyclic Coordinate Descent* (CCD) method [14], and the *Constraint Relaxation* (CR) method [6, 9].

The Jacobian matrix is the matrix of the partial derivatives of the system: $J = \frac{\partial e}{\partial \theta}$. Then $\partial\theta = J^{-1} \cdot \partial e$.

If $e = [e_1 e_2 ... e_N]$ and $\theta = [\theta_1 \theta_2 ... \theta_M]$, then the Jacobian matrix is an $N \times M$ dimension matrix. So, we need a pseudoinverse: $J^+ = (J^T[M \times N] \cdot J[N \times M])^{-1} \cdot J^T[M \times N]$.

The state-changes in the pseudoinverse are minimal, so we get the possible movements in which the relative velocity of the bones in the joints is minimal. Iterative solution: in time moment t_0 all factors are known. The end effector is moved in small steps on the prescribed track, and in each step we calculate the change of state using the Jacobian matrix's pseudoinverse [13].

Algorithm 1: The Jacobian solution

$e = e(t_{start})$;
$\theta = \theta(t_{start})$;
for $t = t_{start}$ **to** t_{end} ***step*** Δt **do**
  Draw the animation if needed ;
  Compute the Jacobian-matrix J;
  Compute the pseudoinverse J^+;
  Compute $e(t + \Delta t)$;
  $\Delta e = e(t + \Delta t) - e(t)$;
  $\Delta\theta = J^+ \cdot \Delta e$;
  $\theta = \theta + \Delta\theta$;
end

For correction we can use the Levenberg–Marquardt algorithm (LMA, LM), also known as the Damped Least-Squares (DLS) method, which is used to solve non-linear least squares problems [20]. These minimization problems arise especially in least squares curve fitting. The essence of the method is that Instead of $\Delta\theta$, we minimize the following expression: $\|J\Delta\theta - \vec{e}\|^2 + \lambda^2 \|\Delta\theta\|^2$, where $\lambda \in \mathbb{R}$ is a damping constant depending on the target position and chosen so that the solution be numerically stable.

The Cyclic Coordinate Descent (CCD) method traces the calculation of θ to vector operations (scalar product, cross product, sin, cos, etc.).

The Constraint Relaxation (CR) algorithm is capable to calculate the exact position of any bone in an arbitrary long kinematic chain, and it is easy to implement in larger dimensions. The kinematic chain of joints without rotational constraints can also be understood as point clouds, with distances between points (definite length bones). The kinematic chain of joints without

rotational constraints can also be understood as point clouds, with distances between points (definite length bones). The essence of the algorithm is to stretch the bones one by one to the target and then restore the original size to get closer to the solution. This involves scaling of vectors.

In conclusion, we used the Damped Least-Squares (DLS) Constraint Relaxation (CR) and methods, which we found to be the most appropriate.

5 The Kinect, the skeleton, and the gestures

Developed by Microsoft, Kinect (initial codenamed Project Natal during development) is a line of motion sensing input devices for Xbox 360 and Xbox One video game consoles (release date: 2010) and Microsoft Windows PCs (release date: 2012) [21]. Based around a webcam-style add-on peripheral, it enables users to control and interact with their game console or personal computer without the need for a game controller, only through a natural user interface using gestures and spoken commands. The skeletal mapping technology shown in 2009 was capable of simultaneously tracking four people, with a feature extraction of 48 skeletal points on a human body at 30 Hz. There are two skeleton versions for Kinect 1 and Kinect 2. The Kinect can track up to six skeletons at one time. According to Figure 3 each of these skeletons has 25 joints.

The Kinect skeleton returns joints not bones [5]. The joints form a point cloud in space:

```
Vector4 skeletonPosition[NUI_SKELETON_POSITION_COUNT];
```

Each joint has 11 properties: color (x, y); depth coordinates (x, y); camera coordinates in homogeneous coordinate system (x, y, z, w); and orientation coordinates in homogeneous coordinate system (x, y, z, w), where w is the divider coordinate. In projective geometry *homogeneous coordinates* or *projective coordinates* were introduced by August Ferdinand Möbius in 1827 for handling the infinity. The color coordinates (x, y) are the coordinates of the joint on the image from the color camera. The depth coordinates (x, y) are the coordinates of the joint on the image from the depth camera. The Kinects camera coordinates use the Kinects infrared sensor to find 3D points of the joints in space. The camera space coordinates are handled differently from the color and depth coordinates. Kinect uses quaternions to deliver joint orientation [8].

Kinect's skeleton can be used for gesture recognition, and gestures can be used to control robots.

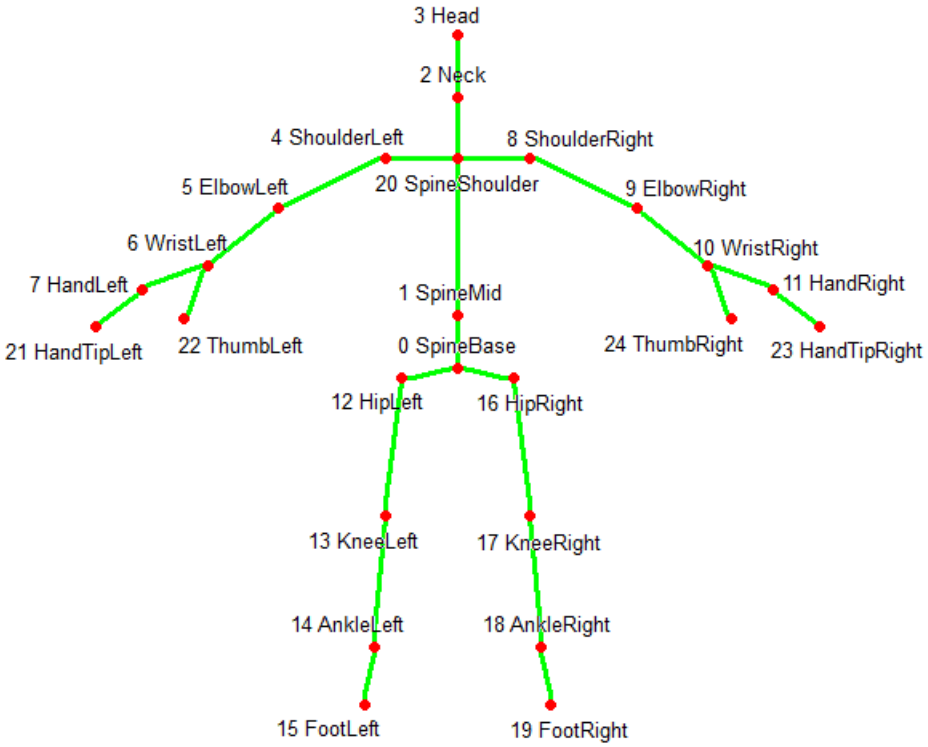

Figure 3: The skeleton of Kinect 2.0 [7]

A gesture is a form of non-verbal communication or non-vocal communication in which visible bodily actions communicate particular messages, either in place of, or in conjunction with, speech. Gestures include movement of the hands, legs, face, or other parts (mostly limbs) of the body. Gesture recognition is a topic in computer science with the goal of interpreting human gestures via mathematical algorithms. We used a 3D model-based skeletal gesture recognition technique.

The method we use assumes that each gesture is represented by 33 frames [2]. We use spherical coordinates, because coordinates are easier to normalize. Regardless of the size of the user, only the distance from the origin will vary, instead all the coordinates in the Cartesian system and the angles will remain constant.

The proposed algorithm for gesture recognition involves 4 steps:

1. detection of the human user,
2. extraction of the features,
3. a stage of warping, in which the gestures are compared to reference gestures,
4. gesture recognition.

Human detection is facilitated by the Kinect sensor, so step 1. is almost trivial. The most important feature of a gesture is limb movement. In step 3. we use a DTW - In time series analysis, *dynamic time warping* (DTW) is one of the algorithms for measuring similarity between two temporal sequences, which may vary in speed. For instance, similarities in walking could be detected using DTW, even if one person was walking faster than the other [2, 22].

The algorithm consists of the following steps:

Algorithm 2: Gesture recognition

```
Detection of the human user;
Extraction of the features;
DTW-warping;
Gesture recognition;
if YES then
    Robot action;
    Learning a new task;
end
```

Gesture recognition requires the following steps:

- geting the 33 frames video sequence
- geting the joint coordinates
- converting to spherical coordinates
- converting to normalized coordinates
- building the vector of features
- DTW-algorithm

DTW compares the sequence obtained of an unknown gesture with one or more reference patterns or templates. With several such templates, the recognition rate will be more big, but the calculation time increases. Having two sequences represented by the time series: $x = x_1, x_2, ..., x_n$, and $y = y_1, y_2, ..., y_n$,

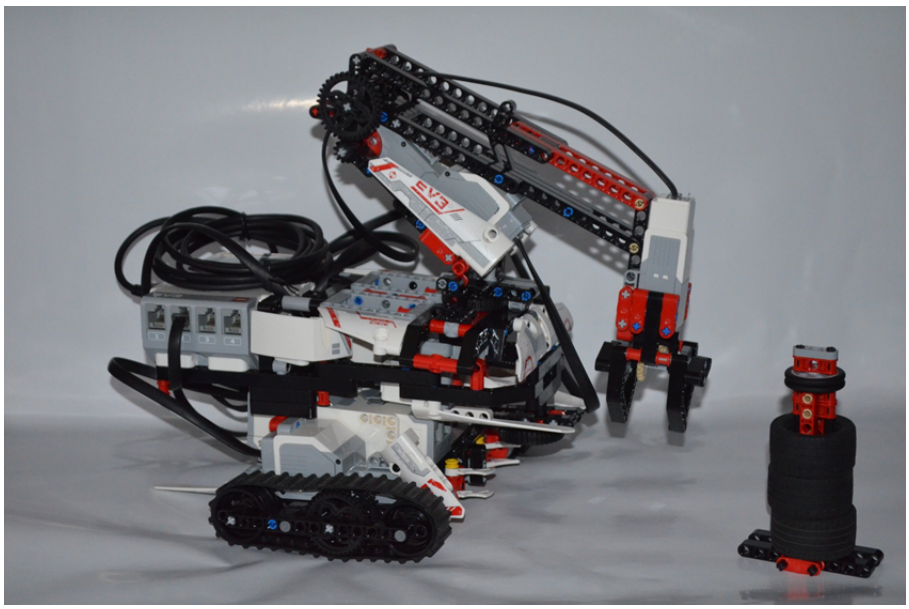

Figure 4: The LEGO robot

a matrix of $n \times m$ elements can be obtained, where each element of the matrix represents the distance between two elements of the time series, called the *cost of the matrix*. To find the best match between these two sequences, a matrix path must be found that minimizes the cumulative total distance between their elements. The *warping path* defines the mapping between the elements of the two time series: $w = w_1, w_2, ..., w_k, ...w_p$, where $DTW(x, y) = \min \sum_{k=1}^{p} d(w_k)$, and $d(w_k)$ represents the distance between elements x_i and y_j of the time series, that is: $d(x_i, y_j) = |x_i - y_j|$.

6 The LEGO robot

Using three interconnected bricks, we built an LEGO MINDSTORMS EV3 [16] robot (Figure 4), which we can control with gestures. Lego Mindstorms EV3 is the third generation robotics kit in Lego's Mindstorms line after NXT (2006) and RCX (1998). The home and education editions were released in 2013. The LEGO MINDSTORMS EV3 set includes motors (large and medium), sensors (touch, color, infrared gyroscopic, ultrasonic), the EV3 programmable brick, cables, more than 550 LEGO Technic elements and a remote control.

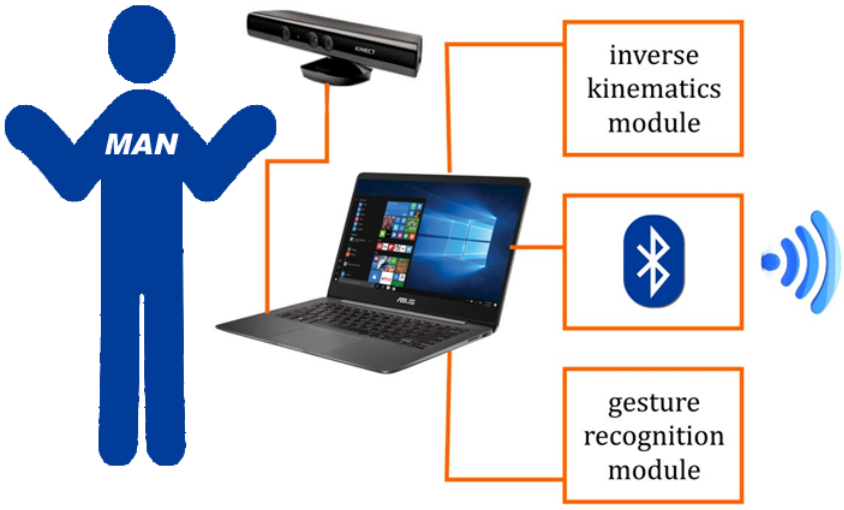

Figure 5: The human part

We have created the system according to Figures 5 and 6, which consists of the following parts:

- Human part:
  - Human user,
  - Kinect,
  - Computer,
  - Inverse kinematics module,
  - Gesture recognition module,
  - Bluetooth communication module.
- Robot part:
  - The robot,
  - Inverse kinematics module,
  - Bluetooth communication module.

The computer-connected Kinect recognizes the user's gestures and calculates the coordinates of the joints using the inverse kinematics module. The data (coordinates, controll sequences) are transmitted via Bluetooth to the LEGO robot. The LEGO robot uses the inverse kinematics module to convert the coordinates into its own coordinate system and then execute the movement.

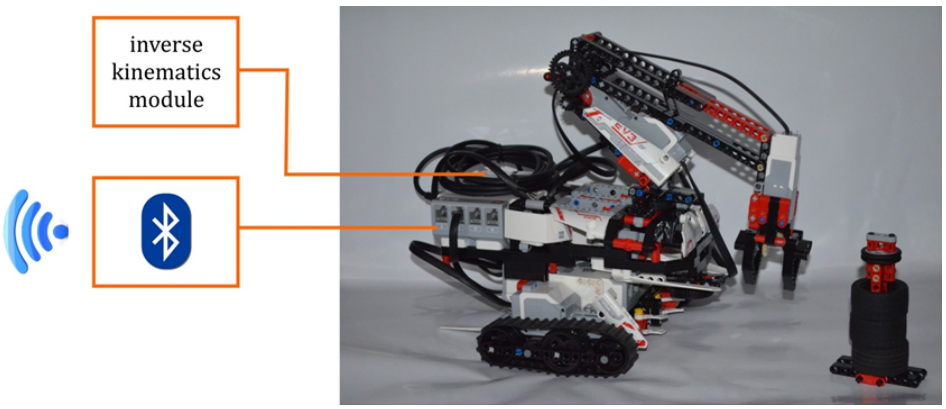

Figure 6: The robot part

7 Conclusion

Experimental results show that the linear approach with the Jacobian solution contains quite a lot of singularity, it must be definitely improved, but it is a fast, effective method. CCD performs better on tracking a moving target avoiding the oscillations and motion discontinuities exhibited by the Jacobian methods [1]. CCD has a low computational cost and solves the IK problem in real-time. The CR algorithm is more difficult to understand but produces suitable results in real time, so it's worth using.

With the presented methods we can easily control robots, of course, we need to practice gestures well enough to achieve the required accuracy.

The methods presented are general enough to control not only LEGO robots, but any robot.

The solution described here is light enough, perfectly suited for educational purposes, and also suitable for lab exercises or simulation exercises.

References

[1] A. Andreas, L. Joan, *Inverse Kinematics: a review of existing techniques and introduction of a new fast iterative solver*, University of Cambridge, Technical Report, 2009. ⇒81, 93

[2] R. G. Boboc, *Natural human-robot interaction for assistive robotics applications*, PhD Thesis, Universitatea Transilvania, Braşov, 2015. ⇒89, 90

[3] T. Brogårdh, Present and future robot control development—An industrial perspective, *Annual Reviews in Control*, **31,** 1 (2017) 69–79. doi:10.1016/j.arcontrol.2007.01.002 ⇒81
[4] S. R. Buss, *Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods*, University of California, San Diego, 2009. ⇒86
[5] L. Jamhoury, *Understanding Kinect V2 Joints and Coordinate System*, Medium, 2018. ⇒88
[6] L. Han, L. Rudolph, *Inverse Kinematics for a Serial Chain with Joints under Distance Constraints*, Clark University Worcester, USA. ⇒86
[7] R. Hoover, *Adventures in Motion Capture: Using Kinect Data*, 2016. ⇒89
[8] L. Jamhoury, *Understanding Kinect V2 Joints and Coordinate System*, 2018. ⇒88
[9] R. Juckett, *Constraint Relaxation IK in 2D*, 2009. ⇒86
[10] L. I. Kovács, *Számítógépes grafika*, Ed. Scientia, Kolozsvár, 2009. ⇒81
[11] L. I. Kovács, *Gesztusokkal vezérelt robotkar*, in: SzámOkt'2017, EMT, Kolozsvár, 2017. ⇒81
[12] N. Sfetcu, *The Art of Movies*, Ebook, 2011. ⇒82
[13] L. Szirmay-Kalos, Gy. Antal , F. Csonka, *Háromdimenziós grafika, animáció és játékfejlesztés*, ComputerBooks, Budapest, 2006. ⇒81, 87
[14] S. J. Wright, *Coordinate Descent Algorithms*, University of Wisconsin, 2010. ⇒86
[15] ∗ ∗ ∗ *Kinect homepage*. ⇒81
[16] ∗ ∗ ∗ *LEGO homepage*. ⇒80, 91
[17] ∗ ∗ ∗ *OpenGL homepage*. ⇒81
[18] ∗ ∗ ∗ *Regression estimation is smoothed values in space as a dynamic movement Motion Selection Principles in Action Robo*. ⇒83
[19] ∗ ∗ ∗ *Wikipedia: Key frame*. ⇒82
[20] ∗ ∗ ∗ *Wikipedia: Levenberg–Marquardt algorithm*. ⇒87
[21] ∗ ∗ ∗ *Wikipedia: Kinect*. ⇒88
[22] ∗ ∗ ∗ *Wikipedia: Dynamic time warping*. ⇒90

Received: June 30, 2019 • Revised: August 6, 2019