VOL. 16, NO. 20, OCTOBER 2021 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2021 Asian Research Publishing Network (ARPN). All rights reserved.

www.arpnjournals.com

EXTRINSIC CAMERA CALIBRATION AND INVERSE KINEMATICS CALCULATION THROUGH ROOT FINDING PROBLEM

Julián E. Herrera B, Robinson Jimenez-Moreno and Jorge A. Aponte-Rodríguez
Faculty of Engineering, Universidad Militar Nueva Granada, Bogotá, Colombia
E-Mail: u3900256@unimilitar.edu.co

ABSTRACT

This paper presents two novel methodological approaches to calculate the extrinsic parameters of cameras and the inverse kinematics of robots, two aspects commonly addressed in autonomous robotic operation that require performing three-dimensional (3D) reconstruction and object gripping. The functionality of these new approaches is demonstrated, and a comparison with current methodologies is performed. The first methodology detail how to calibrate the extrinsic parameters of a camera from linear regression, while the second explains how to find the inverse kinematics using the Newton-Raphson numerical method, also, a metric is presented to define the error between two homogeneous matrices. With this, we can show that it is possible to obtain a matrix of extrinsic parameters with an error of $1.788x10^{-7}$, and observe that the calculation of the inverse kinematics with the proposed method reduces the error on average by 99.8% concerning the conventional way.

Keywords: extrinsic parameters, camera calibration, linear regression, roots, inverse kinematics.

INTRODUCTION

In robotics, two fundamental problems must be solved to control a robot in its operating environment. The inverse kinematics is the first problem, which involves calculating the generalized coordinates based on the desired position and orientation [1], while the second problem involves analyzing the environment using RGB-D cameras to determine positions of interest where the robot must be directed to execute grip tasks [2]. This analysis requires obtaining point clouds from the environment, which is related to the intrinsic and extrinsic parameters of the camera [3] [4]. Currently, several kinds of research have addressed both the analysis of inverse kinematics and the calculation of extrinsic and intrinsic camera parameters. Those addressing inverse kinematics are mostly based on numerical methods, where the error must be reduced to zero by iteratively calculating the value of the generalized coordinates, and researches focused on calculating camera parameters rely on algebraic tools to solve a homogeneous system avoiding trivial solutions.

Calculation of inverse kinematics can be carried out using the algebraic method [5] or the geometric method together with the kinematic decoupling [6]. These generally work for non-redundant robots since they can be described with simple equations. In order to model mechanisms with more than six degrees of freedom, is necessary to apply iterative methods such as Cyclic Coordinate Descent (CCD) [7], which moves one joint at a time to position the gripper on a line that is projected from the desired position down to the joint that is in motion, reducing the Euclidean distance between the gripper and the desired position. However, the final gripper orientation is not considered, so to orient parts, CCD method is not functional. There are iterative methods supported by the robot differential kinematics and its Jacobian [8], where the joint speeds are related to the gripper linear and angular velocity to approximate a solution. Prempraneerach [9] presents an example of the inverse kinematics using the pseudo-inverse Jacobian to control the OWI-535 robot and its five (5) degrees of freedom. This research mentions singularity problems that can be avoided performing variations to the pseudo inverse expression based on statistical concepts such as Damped Least Squares (DLS) [10]. DLS adds a damping factor obtaining more stability to a sudden change of the angular position of the joints. Iterative methods are a suitable approach to tackle a great dimensional problem, such as inverse kinematic of redundant robots. The orientation and positioning of a gripper is a problem that does not have an exact solution, so it must be approximate.

Camera calibration is generally performed, assuming a projection matrix to describe it [11], where cartesian coordinates are mapped to the image plane. This model is usually known as "Pinhole Model" or "Finite Projective Camera" [12], and its parameters are called intrinsic and extrinsic camera parameters. The intrinsic parameters matrix contains the image projection, and the extrinsic parameters matrix describes the camera position and orientation respect to a reference frame, allowing the data captured by an RGB-D camera to be related respect to a coordinate system of interest.

To estimate the camera model, there are two well-known methods used to propose and solve a homogeneous linear system: Direct Linear Transform and Zhang's method. (DLT) method [13] algebraically manipulates the camera model to describe a homogeneous linear equations system, where the unknown variables are all the elements that make up the camera model. Přibyl, B [14] uses this method to estimate the position from line correspondences, solving the homogeneous system as a minimization problem by using least squares to find a vector that is not the trivial solution. Since DLT requires knowing the points in space that are being projected, "Zhang's method" [15] was developed, in which all reference points lie on a plane where it is easier to measure them. The camera calibration is achieved by using a chessboard, where the points are the intersections of the squares, making their coordinates less complex to calculate.

In this work, the inverse kinematics problem [16] and the extrinsic parameters camera calibration are treated with a different approach, demonstrating that they can be solved by implementing the Newton-Raphson numerical method for inverse kinematics and linear regression for calibration of the extrinsic camera parameters. As is described below, methods fully satisfy the problems raised; however, they have limitations that must be considered before being implemented. This work, in addition to presenting the methods, also describes how to implement them in an object manipulation work using an anthropomorphic robot. In order to validate the method, a simulation environment was built in the "CoppeliaSim" software, where an object grasping task was successfully executed.

This paper is divided into six sections, including this introduction. Second section contextualizes the problem handled. In the third section, DLT method is briefly explained, and the proposed method is presented highlighting the differences between them. The inverse kinematics calculation using the conventional and the proposed method is presented in the fourth section detailing the differences between them. The fifth section explains how to apply the methods in an object manipulation task and expose the results obtained. Finally, the sixth section presents work conclusions.

PROBLEM FORMULATION

When a grip task using a manipulator is required, modeling and analysis of the robot environment must be performed. It is necessary to characterize the position and orientation of some entities, such as the gripper of the robot and the objects to be manipulated, in addition, the robot configurations that allow grasping and depositing objects must be calculated. Figure-1 shows an example of an environment where these two problems arise. The image shows an environment with a robot, a camera and a conveyor belt with objects that must be manipulated. To grab objects, the robot must analyze the point cloud given by the camera to define a coordinate system to each object that can be taken as a reference to grabbing them. Then the gripper's coordinate system must be placed into the object's coordinate system using inverse kinematics to finally activate the gripper to hold it. Once the object is attached to the end effector, it can be manipulated to deposit in a specific position.

It is natural to wonder how to get the camera's homogeneous matrix, how to define the camera model to get the point clouds, and how to move the gripper to grab objects. This paper addresses these problems by presenting methodologies to define the camera model and calculate the inverse kinematics. Also, it is explained how to apply them in objects manipulation tasks.

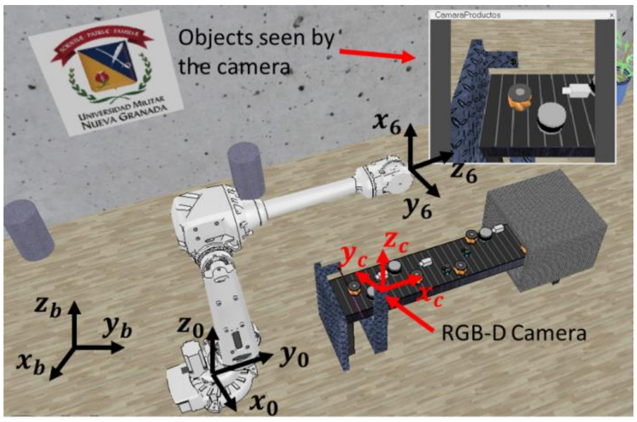

Figure-1. Problem illustration.

EXTRINSIC CAMERA CALIBRATION

Cameras can be described using a perspective projection (Frustum projection [17]) model called "PinHole" [12] (see Figure-2). In this model, a coordinate system is attached to the camera and is referred to using a homogeneous transformation matrix with respect to the inertial system (attached to the robot for this case). Each point that is defined with respect to the inertial coordinate system is projected onto the image plane in (p_x, p_y) coordinates. The mathematical description of these projections is presented in equations (1, 2). P vector, in homogeneous coordinates, is referred to the camera coordinate system using the homogeneous matrix H (see equation (1)). Then, a projection onto the image plane is carried out using the matrix of intrinsic parameters K (equation (2)), generating a vector that is again in homogeneous coordinates. The goal of camera calibration is to calculate the H and K matrices. Generally, the problem is represented by a single matrix (equation (3)), decomposed into the matrices of interest after being estimated.

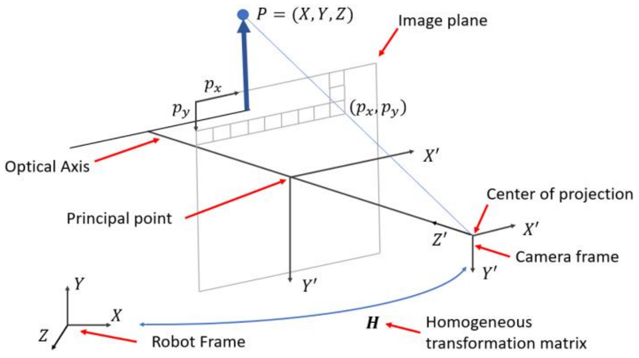

Figure-2. Pinhole model representation.

$$\mathbf{P}' = \begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \mathbf{H}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{HP} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}\mathbf{P} \quad (1)$$

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \begin{bmatrix} p_x Z' \\ p_y Z' \\ Z' \end{bmatrix} = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_x & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \mathbf{KP}' \quad (2)$$

$$\mathbf{p} = \mathbf{TP} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}]\mathbf{P} = \mathbf{KR}[\mathbf{I} \mid -\mathbf{X_o}]\mathbf{P} \quad (3)$$

Zhang's method is widely used to calibrate cameras. It is generally adjusted with a chessboard where the inertial coordinate system is attached to one end of the square, and the intersections of the squares are subsequently measured to use them as the points for the model calibration. This method uses at least 4 points to find the matrix **T**, which contains a homogeneous matrix that relates the coordinate system of the camera to the coordinate system of the chessboard. Because the relationship has not been calculated between the camera frame and the robot frame, this method cannot be applied.

There is another method called “Direct Linear Transform” (DLT). It uses a minimum of 6 points that must not lie on a plane, which makes it more flexible when implementing a robot to generate points in space. This method requires obtaining all point measurements with respect to the robot frame and the image plane. Then it solves a homogeneous system to obtain the camera parameters.

In order to obtain all required points, a sphere is attached to the gripper and used as a reference (see Figure-3). Its position in the inertial frame can be found using the direct kinematics of the robot, while its position in the image plane can be found using morphological filters or detection networks. To get six points, the robot must move to 6 different configurations and then measure sphere position.

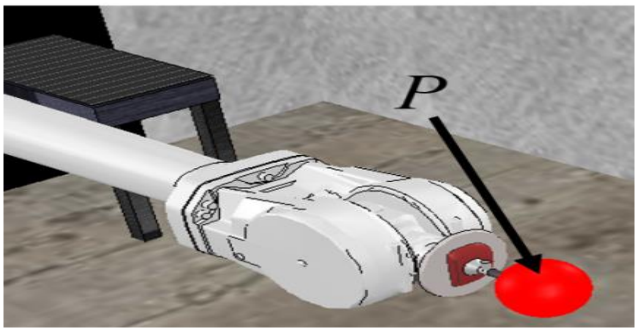

Figure-3. Sphere attached to the gripper used to capture the points for the camera calibration.

DLT method separates the matrix **T** into three vectors A, B and C. Equations system presented in (4) is algebraically manipulated to express the problem (5) using the vectors shown in (6) and (7). Equation (5) considers one point in space, and the remain 5 points required by this method are added according to (8) and (9). Equation (9) is not equal to zero since a contradictions vector is generated by error measurement W. In order to reduce the error, a minimization problem is proposed such that a non-zero matrix T is found reducing the Ω value (see equation (10)). This minimization problem can be solved using a Singular Value Decomposition (SVD) [18] for the matrix M. Decomposition by singular values will result in three matrices like the ones shown in (11), where the solution will be given by column V that belongs to the smallest singular value of the matrix S. After the matrix **T** is estimated, vector $\mathbf{X_o}$ can be calculated as in equation (12). In equation (13), a QR decomposition is performed to find the rotation matrix and the intrinsic parameters matrix.

$$\mathbf{T} = \begin{bmatrix} t_{1,1} & t_{1,2} & t_{1,3} & t_{1,4} \\ t_{2,1} & t_{2,2} & t_{2,3} & t_{2,4} \\ t_{3,1} & t_{3,2} & t_{3,3} & t_{3,4} \end{bmatrix} == \begin{bmatrix} \mathbf{A}^T \\ \mathbf{B}^T \\ \mathbf{C}^T \end{bmatrix} \quad (4)$$

$$\begin{bmatrix} \mathbf{a}_x^T \\ \mathbf{a}_y^T \end{bmatrix} \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{C} \end{bmatrix} = \mathbf{0} \quad (5)$$

$$\mathbf{a}_x^T = [-X \quad -Y \quad -Z \quad -1 \quad 0 \quad 0 \quad 0 \quad 0 \quad p_x X \quad p_x Y \quad p_x Z \quad p_x] \quad (6)$$

$$\mathbf{a}_y^T = [0 \quad 0 \quad 0 \quad 0 \quad -X \quad -Y \quad -Z \quad -1 \quad p_y X \quad p_y Y \quad p_y Z \quad p_y] \quad (7)$$

$$\mathbf{M} = \begin{bmatrix} \mathbf{a}_{x1}^T \\ \mathbf{a}_{y1}^T \\ \mathbf{a}_{x2}^T \\ \mathbf{a}_{y2}^T \\ \vdots \\ \mathbf{a}_{xn}^T \\ \mathbf{a}_{yn}^T \end{bmatrix} \quad (8)$$

$$\mathbf{M} \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{C} \end{bmatrix} = \mathbf{W} \quad (9)$$

$$\Omega = \mathbf{W}^T \mathbf{W} \quad (10)$$

$$\mathbf{M} = \mathbf{USV}^T \quad (11)$$

$$\mathbf{X_o} = -\mathbf{T}_{1:3,1:3}^{-1} \mathbf{T}_{1:3,4} \quad (12)$$

$$\mathbf{T}_{1:3,1:3}^{-1} = \mathbf{R}^{-1} \mathbf{K}^{-1} \quad (13)$$

DLT method is not iterative, and it requires solving a homogeneous system as a minimization problem using SVD and extract the interest matrices. In the proposed method, it is possible to calculate the extrinsic parameters matrix through a linear regression obtained

VOL. 16, NO. 20, OCTOBER 2021 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2021 Asian Research Publishing Network (ARPN). All rights reserved.

www.arpnjournals.com

from a minimization problem. The method developed here is useful for problems that do not require calibration of intrinsic parameters, for example, in the case of previously calibrated cameras or in situations where it is desired to find the homogeneous matrix that relates the coordinate systems of various manipulators. The method is presented below, focusing on finding the homogeneous matrix that relates a camera and a robot.

Initially, several points must be measured in two coordinate systems of interest using as reference the sphere attached to the gripper. The position of the sphere regarding the camera is obtained by finding P´ in equation (2) considering the deep found by RGB-D camera of analysed pixel. If 4 different positions of the sphere are measured, the problem can be represented as shown in Figure-5. There are two coordinate systems a and b with a known points p_0 to p_3. Equations (14) and (15) can be expressed if an auxiliary coordinate system r is assigned to the point p_0 which has the same orientation of the a coordinate system. In equation (14), homogeneous matrix is composed of an identity matrix and a translation vector, point p_0, measured with respect to the coordinate system **a**. Equation (15) is applied to find the matrix of interest ${}^a\mathbf{T}_b$. If the ${}^b\mathbf{T}_r$ matrix is estimated, it is possible to solve the problem.

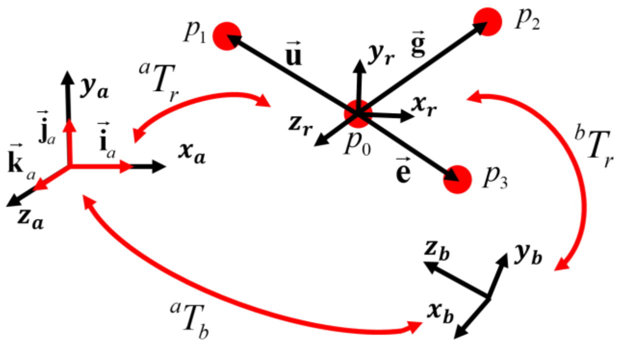

Figure-4. Coordinate systems of interest related by homogeneous matrices.

$$ {}^a\mathbf{T}_r = \begin{bmatrix} \mathbf{I} & {}^a\mathbf{p}_0 \\ \mathbf{0} & 1 \end{bmatrix} \qquad (14)$$

$$ {}^a\mathbf{T}_b = {}^a\mathbf{T}_r \left({}^b\mathbf{T}_r\right)^{-1} \qquad (15)$$

Each coordinate system has three unitary vectors projected onto each coordinate axis. ${}^b\mathbf{T}_r$ matrix can be calculated if the vectors specified in (16) are known and referred to the **b** system. As shown below, the unknown vectors ${}^b\vec{\mathbf{i}}_r$ and ${}^b\vec{\mathbf{j}}_r$ can be estimated using a linear regression. For this purpose dot product with auxiliary vectors **u**, **g** and **e** is used (Figure-4). These vectors are measured from the **r** coordinate system and they became unitary through equations (17) to (19). Since the orientation of systems **r** and **a** are the same, is valid to use points measured regarding system **a.** Components of unitary vectors represent cosines directors of vector **u, g** and **e** used in dot product with vector ${}^b\vec{\mathbf{i}}_r$, ${}^b\vec{\mathbf{j}}_r$ and ${}^b\vec{\mathbf{k}}_r$.

$$ {}^b\mathbf{T}_r = \begin{bmatrix} {}^b\mathbf{i}_r & {}^b\mathbf{j}_r & {}^b\mathbf{i}_r \times {}^b\mathbf{j}_r & {}^b\mathbf{p}_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (16)$$

$$ \mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \frac{{}^a\mathbf{u}}{\left|{}^a\mathbf{u}\right|} = \frac{{}^a\mathbf{p}_1 - {}^a\mathbf{p}_0}{\left|{}^a\mathbf{p}_1 - {}^a\mathbf{p}_0\right|} \qquad (17)$$

$$ \mathbf{c} = \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} = \frac{{}^a\mathbf{g}}{\left|{}^a\mathbf{g}\right|} = \frac{{}^a\mathbf{p}_2 - {}^a\mathbf{p}_0}{\left|{}^a\mathbf{p}_2 - {}^a\mathbf{p}_0\right|} \qquad (18)$$

$$ \mathbf{d} = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} = \frac{{}^a\mathbf{e}}{\left|{}^a\mathbf{e}\right|} = \frac{{}^a\mathbf{p}_3 - {}^a\mathbf{p}_0}{\left|{}^a\mathbf{p}_3 - {}^a\mathbf{p}_0\right|} \qquad (19)$$

Once the cosine of the angle of **u**, **g** and **e** vectors respect to ${}^b\vec{\mathbf{i}}_r$, ${}^b\vec{\mathbf{j}}_r$ and ${}^b\vec{\mathbf{k}}_r$ vectors are known, the dot products indicated in equations (20) and (21) can be proposed, these represent a set of equations where the unknowns are the vectors to use in the rotation matrix of equation (16). Equations (20) and (21) are a linear model that can be written as indicated in equation (22). **X** must be understood as the inputs of the system, **Y** as the expected values and the **β** as the coefficients that must be estimated. There is a random error **e** assumed with a normal distribution with zero mean and constant variance. Approximate model of (22) is presented in (23), which is solved through a minimization problem in which the $\hat{\boldsymbol{\beta}}$ estimators must be found, reducing the sum of the squared errors between expected and estimated values (see equation (24)). The solution of this system is presented in equation (25), which implements a left pseudo-inverse.

$$ \begin{bmatrix} \left|{}^b\mathbf{u}\right| v_x \\ \left|{}^b\mathbf{g}\right| c_x \\ \left|{}^b\mathbf{e}\right| d_x \end{bmatrix} = \begin{bmatrix} {}^b\vec{\mathbf{i}}_r \cdot {}^b\mathbf{u} \\ {}^b\vec{\mathbf{i}}_r \cdot {}^b\mathbf{g} \\ {}^b\vec{\mathbf{i}}_r \cdot {}^b\mathbf{e} \end{bmatrix} = \begin{bmatrix} {}^b\mathbf{u}^T \\ {}^b\mathbf{g}^T \\ {}^b\mathbf{e}^T \end{bmatrix} {}^b\vec{\mathbf{i}}_r \qquad (20)$$

$$ \begin{bmatrix} \left|{}^b\mathbf{u}\right| v_y \\ \left|{}^b\mathbf{g}\right| c_y \\ \left|{}^b\mathbf{e}\right| d_y \end{bmatrix} = \begin{bmatrix} {}^b\vec{\mathbf{j}}_r \cdot {}^b\mathbf{u} \\ {}^b\vec{\mathbf{j}}_r \cdot {}^b\mathbf{g} \\ {}^b\vec{\mathbf{j}}_r \cdot {}^b\mathbf{e} \end{bmatrix} = \begin{bmatrix} {}^b\mathbf{u}^T \\ {}^b\mathbf{g}^T \\ {}^b\mathbf{e}^T \end{bmatrix} {}^b\vec{\mathbf{j}}_r \qquad (21)$$

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e} \tag{22}$$

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\boldsymbol{\beta}} \tag{23}$$

$$\hat{\boldsymbol{\beta}} = \arg\min_{\hat{\boldsymbol{\beta}} \in \mathbb{R}^3} \left(\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right)^T \left(\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right) \tag{24}$$

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}^T\mathbf{Y} \tag{25}$$

In the proposed methodology, it is necessary to know at least 4 points in space that can be measured respect to the two coordinate interest systems, where one can belong to an RGB-D camera. In this method, the estimation of parameters from equation (43) are based in a ${}^{b}\mathbf{p}_0$ vector, which is the base to perform all calculus. Final result depends largely on the accuracy of ${}^{b}\mathbf{p}_0$. DLT method unlike the explained method takes into account all the points, but it does not rely on one to carry out calculations. It is also important to note that the DLT method calculates intrinsic parameters at the same time, while the explained method can only be used to calculate the homogeneous matrix that relates two systems, not a projection matrix. Both methods estimate the parameters using a minimization problem; however, DLT method obtains a homogeneous system with the drawback of having the trivial solution.

Matrix of intrinsic and extrinsic parameters allows to use equation (1) and (2) to perform the reconstruction of the environment from the information of each pixel coordinate and its depth. If a detection system indicates that some pixels belong to an object, its cartesian coordinates can be found, and the robot can be moved to it. To move the robot, a method that calculates the state vector of the robot is required, below inverse kinematics methods are presented.

INVERSE KINEMATICS CALCULATION

The classic method for iteratively calculating the robot inverse kinematics is based on updating the state vector $\mathbf{q}$ as shown in equation (26), where the updated value of the state vector $\mathbf{q}_{k+1}$ is equal to its current $\mathbf{q}_k$ value plus a change defined by the speed vector $\mathbf{W}_k$, the pseudo inverse of the Jacobian matrix $\mathbf{J}^+$ and a weight vector $\boldsymbol{\alpha}$ that controls the joints speed change. Equation (26) is given by the robot's differential kinematics where the angular and joint speeds are related. To apply the method, it is necessary to calculate a velocity vector reducing the error between the gripper coordinate system (see equation (27)) and the desired one (see equation (28)). Matrix ${}^{b}\mathbf{T}_6$ describes the coordinate system 6 referred to the base system.

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \Delta\mathbf{q} = \mathbf{q}_k + \boldsymbol{\alpha}^T \left(\mathbf{J}^+\mathbf{W}_k\right) \tag{26}$$

$${}^{b}\mathbf{T}_n = \begin{bmatrix} {}^{b}\mathbf{n}_n & {}^{b}\mathbf{o}_n & {}^{b}\mathbf{a}_n & {}^{b}\mathbf{t}_n \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^{b}\mathbf{T}_6 \tag{27}$$

$${}^{b}\mathbf{T}_d = \begin{bmatrix} {}^{b}\mathbf{n}_d & {}^{b}\mathbf{o}_d & {}^{b}\mathbf{a}_d & {}^{b}\mathbf{t}_d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{28}$$

In order to reduce the error between the coordinate systems, their distance is reduced as the vectors of the rotation matrix are oriented. This is achieved if the velocity vector is defined according to equation (29), where the linear velocity vector $\mathbf{d}$ is the position error and the angular velocity vector $\boldsymbol{\omega}$ is defined by a sum of cross products (Rodríguez's formula [16]). Equation (29) results are used in equation (26) iteratively returning the future state vector to the current one, finding the generalized coordinates value and therefore the required robot configuration.

$$\mathbf{W}_k = \begin{bmatrix} \mathbf{d} \\ \boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} {}^{b}\mathbf{t}_d - {}^{b}\mathbf{t}_n \\ {}^{b}\mathbf{n}_n \times {}^{b}\mathbf{n}_d + {}^{b}\mathbf{o}_n \times {}^{b}\mathbf{o}_d + {}^{b}\mathbf{a}_n \times {}^{b}\mathbf{a}_d \end{bmatrix} \tag{29}$$

Equation (30) allows to calculate the Jacobian [16], it has a number of vectors defined by the Degrees of Freedom (DOF). If a vector of the Jacobina represents a rotational joint, equation (31) must be used to define it, if it represents a prismatic joint, equation (32) must be used. Vector ${}^{b}\mathbf{z}_{i\text{-}1}$ is oriented according to the rotational or prismatic joint axis using the Denavith - Hartemberg (DH) convention. Superscript is used to reference the vector respect to the base frame and subscript is related with the joint that the vector represents. A detailed explanation can be found in [16].

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_0 & \mathbf{J}_1 & \cdots & \mathbf{J}_{n-1} \end{bmatrix} \tag{30}$$

$$\mathbf{J}_i = \begin{bmatrix} {}^{b}\mathbf{z}_{i\text{-}1} \times \left({}^{b}\mathbf{t}_n - {}^{b}\mathbf{t}_{i\text{-}1}\right) \\ {}^{b}\mathbf{z}_{i\text{-}1} \end{bmatrix} \tag{31}$$

$$\mathbf{J}_i = \begin{bmatrix} {}^{b}\mathbf{z}_{i\text{-}1} \\ \mathbf{0} \end{bmatrix} \tag{32}$$

The approach proposed is based on applying the Newton-Raphson numerical method to a series of equations directly related to the robot direct kinematics and the derivatives of its matrices. The functions found do not represent the robot kinematics model, unlike the conventional method; additionally, the Jacobian found is obtained from the same function to which the roots are being found. Results show smaller errors using the proposed method than those obtained with the classical method.

In order to explain the proposed method, the Newton - Raphson numerical method for a Multivariable Vector Function (MVF) is initially defined. In equation

(33) roots for the nonlinear MVF $f(\mathbf{X})$ are found. This function depends on a $\mathbf{X}$ vector (see equation 34) where the generalized coordinates for a six DOF robot are represented. Taylor series are used to approximate $f(\mathbf{X})$ as a linearized function around the $\mathbf{a}$ operation point as is shown in equation (35). In equation (36), vector $\mathbf{X}$ is obtained by assigning a zero value to $f(\mathbf{X})$, this equation is iteratively used to find the system roots (see equation. (37)). To avoid invertibility problems, the Jacobian inverse is changed for its pseudoinverse.

$$f(\mathbf{X}) = \begin{bmatrix} f_1(\mathbf{X}) \\ f_2(\mathbf{X}) \\ \vdots \\ f_N(\mathbf{X}) \end{bmatrix} \tag{33}$$

$$\mathbf{X} = [x_1, x_2, \cdots, x_n]^T = \mathbf{q} = [\mathrm{q}_0, \mathrm{q}_1, \mathrm{q}_2, \mathrm{q}_3, \mathrm{q}_4, \mathrm{q}_5]^T \tag{34}$$

$$f(\mathbf{X}) \approx \begin{bmatrix} f_1(\mathbf{a}) \\ f_2(\mathbf{a}) \\ \vdots \\ f_N(\mathbf{a}) \end{bmatrix} + \begin{bmatrix} \frac{\partial f_1(\mathbf{a})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{a})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_N(\mathbf{a})}{\partial x_1} & \cdots & \frac{\partial f_N(\mathbf{a})}{\partial x_n} \end{bmatrix} [\mathbf{X} - \mathbf{a}] \tag{35}$$
$$= f(\mathbf{a}) + \mathbf{J}[\mathbf{X} - \mathbf{a}]$$

$$\mathbf{X} = \mathbf{a} - (\mathbf{J}(\mathbf{a}))^{-1} f(\mathbf{a}) \tag{36}$$

$$\mathbf{X}_{i+1} = \mathbf{X}_i - (\mathbf{J}(\mathbf{X}_i))^{+} f(\mathbf{X}_i) \tag{37}$$

To apply the previous method, a vector function that analyses the error between two homogeneous matrices is proposed (see equation (38)). If function norm is zero, then two matrices must be equal. This is a metric proposed to evaluate the error.

$$F(\mathbf{X}) = \begin{bmatrix} F_{pos}(\mathbf{X}) \\ F_{orient}(\mathbf{X}) \end{bmatrix} = \begin{bmatrix} {}^b\mathbf{t}_d - {}^b\mathbf{t}_n \\ -1 + {}^b\mathbf{n}_n \cdot {}^b\mathbf{n}_d \\ -1 + {}^b\mathbf{o}_n \cdot {}^b\mathbf{o}_d \\ -1 + {}^b\mathbf{a}_n \cdot {}^b\mathbf{a}_d \end{bmatrix} \tag{38}$$

$F_{pos}(\mathbf{X})$ represents the position error and $F_{orient}(\mathbf{X})$ contains the dot product error between the rotation matrices vectors. When it is necessary to orient only one axis of the rotation matrices, use the dot product error of the analyzed axis. To orient all axes, use only the dot product error of two axis.

For equation (35) is necessary to calculate the gradient of each function to generate the Jacobian. For this purpose, equation (27) is symbolically described (see eq. (39)) to have the interest vectors expressed as a function of the state vector to later calculate the symbolic Jacobian. A problem with this approach is presented due to the high computational cost involved in the calculation of the symbolic Jacobian. To avoid this problem, partial derivatives are analysed as shown in equations (40) and (41).

$${}^b\mathbf{T}_n(\mathbf{q}) = \begin{bmatrix} {}^b\mathbf{n}_n(\mathbf{q}) & {}^b\mathbf{o}_n(\mathbf{q}) & {}^b\mathbf{a}_n(\mathbf{q}) & {}^b\mathbf{t}_n(\mathbf{q}) \\ 0 & 0 & 0 & 1 \end{bmatrix} = \tag{39}$$
$${}^b\mathbf{T}_0\,{}^0\mathbf{T}(q_0)_1\,{}^1\mathbf{T}_2(q_1)\,{}^2\mathbf{T}_3(q_2)\,{}^3\mathbf{T}_4(q_3)\,{}^4\mathbf{T}_5(q_4)\,{}^5\mathbf{T}_6(q_5)$$

$$\frac{\partial F_{pos}(\mathbf{q})}{\partial \mathbf{q}} = \frac{\partial({}^b\mathbf{t}_d - {}^b\mathbf{t}_n)}{\partial \mathbf{q}} = \frac{-\partial\,{}^b\mathbf{t}_n}{\partial \mathbf{q}} = \tag{40}$$
$$-\begin{bmatrix} \frac{\partial\,{}^b\mathbf{t}_n}{\partial q_0} & \frac{\partial\,{}^b\mathbf{t}_n}{\partial q_1} & \cdots & \frac{\partial\,{}^b\mathbf{t}_n}{\partial q_{n-1}} \end{bmatrix}$$

$$\frac{\partial F_{orient}(\mathbf{X})}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial\left(-1 + {}^b\mathbf{n}_n \cdot {}^b\mathbf{n}_d\right)}{\partial \mathbf{q}} \\ \frac{\partial\left(-1 + {}^b\mathbf{o}_n \cdot {}^b\mathbf{o}_d\right)}{\partial \mathbf{q}} \\ \frac{\partial\left(-1 + {}^b\mathbf{a}_n \cdot {}^b\mathbf{a}_d\right)}{\partial \mathbf{q}} \end{bmatrix} = \tag{41}$$
$$\begin{bmatrix} \frac{\partial\,{}^b\mathbf{n}_n}{\partial q_0} \cdot {}^b\mathbf{n}_d & \frac{\partial\,{}^b\mathbf{n}_n}{\partial q_1} \cdot {}^b\mathbf{n}_d & \cdots & \frac{\partial\,{}^b\mathbf{n}_n}{\partial q_{n-1}} \cdot {}^b\mathbf{n}_d \\ \frac{\partial\,{}^b\mathbf{o}_n}{\partial q_0} \cdot {}^b\mathbf{o}_d & \frac{\partial\,{}^b\mathbf{o}_n}{\partial q_1} \cdot {}^b\mathbf{o}_d & \cdots & \frac{\partial\,{}^b\mathbf{o}_n}{\partial q_{n-1}} \cdot {}^b\mathbf{o}_d \\ \frac{\partial\,{}^b\mathbf{a}_n}{\partial q_0} \cdot {}^b\mathbf{a}_d & \frac{\partial\,{}^b\mathbf{a}_n}{\partial q_1} \cdot {}^b\mathbf{a}_d & \cdots & \frac{\partial\,{}^b\mathbf{a}_n}{\partial q_{n-1}} \cdot {}^b\mathbf{a}_d \end{bmatrix}$$

Resulting derivatives are found as shown in equation (42) which are simplified in equation (43). The result indicates that in order to calculate the partial vectors derivatives of the homogeneous matrix ${}^b\mathbf{T}_n$ regarding a component of the state vector $\mathbf{q}$, it is necessary to use the direct robot kinematics changing only the matrix containing the analyzed state vector component for its partial derivative of the same component. Thus, each column in equations (40) and (41) can be filled building the Jacobian.

$$\frac{\partial\,{}^b\mathbf{T}_n}{\partial q_{i-1}} = \begin{bmatrix} \frac{\partial\,{}^b\mathbf{n}_n}{\partial q_{i-1}} & \frac{\partial\,{}^b\mathbf{o}_n}{\partial q_{i-1}} & \frac{\partial\,{}^b\mathbf{a}_n}{\partial q_{i-1}} & \frac{\partial\,{}^b\mathbf{t}_n}{\partial q_{i-1}} \\ 0 & 0 & 0 & 0 \end{bmatrix} = \tag{42}$$
$$\frac{\partial\left({}^b\mathbf{T}_{i-1}\,{}^{i-1}\mathbf{T}(q_{i-1})_i\,{}^i\mathbf{T}_n\right)}{\partial q_{i-1}}$$

$$\frac{\partial\,{}^b\mathbf{T}_n}{\partial q_{i-1}} = {}^b\mathbf{T}_{i-1}\frac{\partial\,{}^{i-1}\mathbf{T}(q_{i-1})_i}{\partial q_{i-1}}\,{}^i\mathbf{T}_n \tag{43}$$

Both conventional and proposed method iterate over a function that needs an update of the Jacobian for each cycle. The difference between these methods resides in how the Jacobian is defined and the functions at which their roots are found. In the proposed methodology is necessary to know the derivative of a homogeneous transformation regarding the variable that contains it, while the conventional method requires to know the vectors (${}^b\mathbf{z}_{i\text{-}1}, {}^b\mathbf{t}_n, {}^b\mathbf{t}_{i\text{-}1}$) which are referenced to the base frame. Note that in the classic method the Jacobian can be defined by computing the direct kinematics once, while in the proposed method, the Jacobian is defined by iterating over the matrices of direct kinematics, so the proposed is expected to be slower.

RESULTS

CoppeliaSim [19] is used to verify the results of the methods previously explained. An ABB IRB 4600 robot is implemented; its DH parameters are presented in Table-1. To calibrate the extrinsic parameters of the camera, four points are measured from the robot frame and camera frame, as shown in Table-2, then equations (20) and (21) are constructed resulting in equations (44) and (45). Subsequently, each vector is determined using linear regression of (25), as is shown in equation (46).

Table-1. DH parameters for the IRB 4600 Robot.

Matriz	θ[deg]	$d[m]$	$a[m]$	α[deg]
${}^b\mathbf{T}_0$	–175.4	0	0.5146	0
${}^0\mathbf{T}_1(q_0)$	$-150.1+q_0$	0.495	0.2122	–90
${}^1\mathbf{T}_2(q_1)$	$-78.1+q_1$	0.3366	1.1191	–34.5
${}^2\mathbf{T}_3(q_2)$	$36.3+\mathrm{q}_2$	–0.5463	0.1616	–90.6
${}^3\mathbf{T}_4(q_3)$	q_3	0.7962	0.8272	95
${}^4\mathbf{T}_5(q_4)$	$-139.6+q_4$	–0.0034	0.0037	93.6
${}^5\mathbf{T}_6(q_6)$	$-176.6+q_6$	0.0006	0.0098	–93.7

Table-2. Captured points to calibrate the extrinsic camera parameters.

Sistema coordenado	$\mathbf{p}_0$	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$
Inercial	${}^b\mathbf{p}_0 = \begin{pmatrix}-16.42x10^{-3}\\-0.733\\2.576\end{pmatrix}$	${}^b\mathbf{p}_1 = \begin{pmatrix}-24.66x10^{-3}\\0.563\\4.042\end{pmatrix}$	${}^b\mathbf{p}_2 = \begin{pmatrix}-22.97x10^{-3}\\0.884\\2.904\end{pmatrix}$	${}^b\mathbf{p}_3 = \begin{pmatrix}-1.121\\-0.915\\2.621\end{pmatrix}$
Cámara	${}^a\mathbf{p}_0 = \begin{pmatrix}-1.498\\-41.45x10^{-3}\\2.04\end{pmatrix}$	${}^a\mathbf{p}_1 = \begin{pmatrix}0.405\\-49.69x10^{-3}\\1.591\end{pmatrix}$	${}^a\mathbf{p}_2 = \begin{pmatrix}15.05x10^{-3}\\-48.009x10^{-3}\\2.708\end{pmatrix}$	${}^a\mathbf{p}_3 = \begin{pmatrix}-1.621\\-1.146\\1.907\end{pmatrix}$

The matrix in (16) is completed with vectors from (46) and is used together with (14) to solve equation (15). The calculated matrix can be seen in equation (47). Using the norm of (38) (omitting the last element of the vector), the error between the obtained matrix from the simulator and the calculated matrix is equal to $1.788x10^{-7}$, proving that the matrices are almost the same. It is expected a bigger error in a real environment due to the camera distortion and measurement errors.

$$\begin{bmatrix}1.903\\1.513\\-0.1226\end{bmatrix} = \begin{bmatrix}-8.24x10^{-3} & 1.297 & 1.466\\ -6.55x10^{-3} & 1.618 & 0.328\\ -1.104 & -0.1816 & 45.53x10^{-3}\end{bmatrix} {}^b\vec{\mathbf{i}}_r \tag{44}$$

$$\begin{bmatrix}-8.24x10^{-3}\\-6.551x10^{-3}\\-1.104\end{bmatrix} = \begin{bmatrix}-8.24x10^{-3} & 1.297 & 1.466\\ -6.55x10^{-3} & 1.618 & 0.328\\ -1.104 & -0.1816 & 45.53x10^{-3}\end{bmatrix} {}^b\vec{\mathbf{j}}_r \tag{45}$$

$$\begin{bmatrix}{}^b\vec{\mathbf{i}}_r^T\\ {}^b\vec{\mathbf{j}}_r^T\end{bmatrix} = \begin{bmatrix}6.74x10^{-9} & 0.819 & 0.5735\\ 0.999 & 2.152x10^{-9} & 1.507x10^{-9}\end{bmatrix} \tag{46}$$

$${}^a\mathbf{T}_b = \begin{bmatrix}6.7x10^{-9} & 0.819 & 0.573 & -2.375\\ 1 & -5.5x10^{-9} & -3.86x10^{-9} & -25.03x10^{-3}\\ 2.1x10^{-14} & 0.5735 & -0.819 & 4.58\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{47}$$

VOL. 16, NO. 20, OCTOBER 2021 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2021 Asian Research Publishing Network (ARPN). All rights reserved.
www.arpnjournals.com

Knowing the image depth channel generated by RGB-D cameras, a point cloud is obtained as shown in Figure-5.

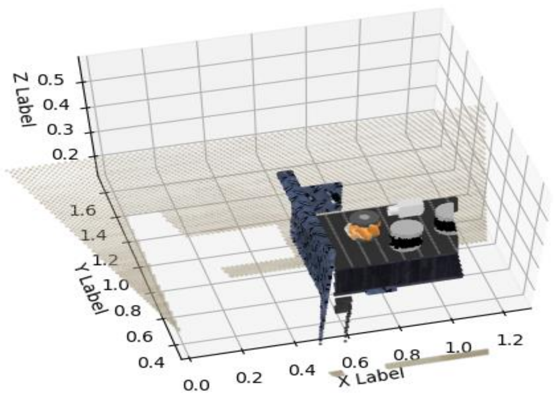

Figure-5. 3D Environment reconstruction.

By using a segmentation system based on deep learning, such as a Mask-RCNN [20], it is possible to filter the information allowing to create the points cloud for the detected objects (see Figure-6).

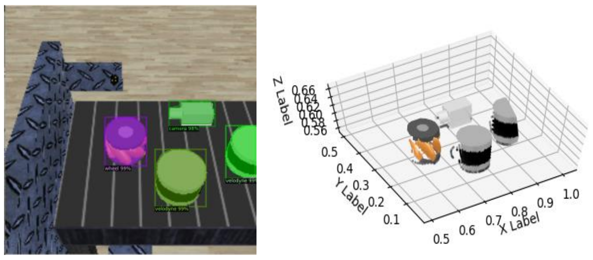

Figure-6. Objects segmentation and its points cloud.

The calculus of inverse kinematics has been tested using both methods explained. In the classical method the weight vector is set to one. Also, the Norm of equation (38) has been used to analyze the total error between the desired homogeneous matrix and the gripper matrix. For the first test, the reference homogeneous matrix was set as presented in equation (48); additionally, a state vector equal to the null vector was used to start the methods. Results obtained after performing the inverse kinematics are shown in Figure-7, considering both the orientation and position of the gripper. In all methods, error variation stabilized close at nineteen iterations.

$$ {}^{b}\mathbf{T}_d = \begin{bmatrix} 0.707 & -0.707 & 0 & -1.2 \\ 0.707 & 0.707 & 0 & -0.041 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{48} $$

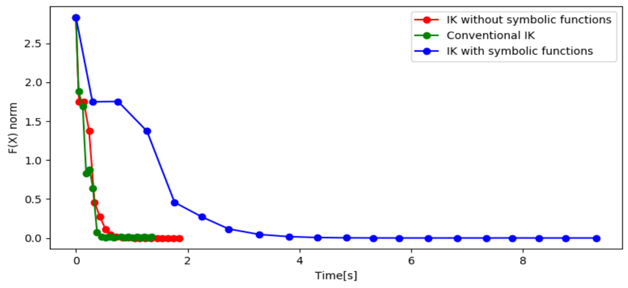

Figure-7. Norm of equation (38) using the three methods explained.

According to Figure-7, there is a significant difference in the execution speed when performing the inverse kinematics for the three analyzed methods. In the inverse kinematics calculation, the Jacobian solution is the most time-consuming task due to its iterative construction. The conventional method achieves to calculate the Jacobian in less time since it uses vectors found by calculating the robot direct kinematics, while the proposed method involves a longer procedure; however, the obtained error is smaller. Symbolic method is the most expensive and time consuming for calculating the Jacobian.

In order to validate the method functionality, a random homogenous matrix is chosen, and its inverse kinematics with its state vector is calculated. The state vector is used to move and to orient the robot in the simulation environment using both the conventional and the proposed method. Then the homogeneous matrix of the gripper is obtained from the simulator and compared with the initially proposed matrix. The procedure is performed five times as shown in table 3. The results show that our method produces smaller errors. On average the proposed method reduces the error by 99, 84% in comparison with the conventional method.

It is important to notice that the explained methods still have singularity problems that should be addressed using a damping factor in the Jacobian pseudoinverse. Otherwise, for some positions, the determinant obtained could be equal to zero preventing the inverse kinematics from being determined.

Table-3. Resulting error between Coppelia homogeneous matrix and desired homogeneous matrix.

Test\ Method	Conventional IK	Proposed IK
1	$78.9x10^{-3}$	$2.26x10^{-5}$
2	$93.03x10^{-3}$	$2.568x10^{-5}$
3	$3.02x10^{-3}$	$7.81x10^{-5}$
4	$1.04x10^{-3}$	$4.18x10^{-5}$
5	$75.3x10^{-3}$	$2.1x10^{-4}$

Through the inverse kinematics exposed, it is possible to direct the robot to the centroid of the segmented object shown in Figure-6. In this way, a satisfactory grip is achieved as presented in Figure-8.

CamaraProductos

(a) (b)

Figure-8. a) Robot gripper located at the calculated centroid. b) Object manipulation.

CONCLUSIONS

This paper presents two methods to calculate both the robot inverse kinematics and the homogeneous matrices relating two coordinated systems. Deductions found are based in the minimization of an objective function where the roots represent the problem solution. Results obtained within the simulation environment validate the methods functioning so they can be used in other developments requiring the calibration of extrinsic parameters and the calculation of the robot generalized coordinates.

High dimensionality problems such as the inverse kinematics can be addressed using the Newton Raphson numerical method. The proposed method achieves higher accuracy calculating the state vector in comparison within the conventional method, although it involves a higher computational cost and longer execution time. Method to be used must be decided considering accuracy and processing time. Jacobian calculation using symbolic functions involves a high computational cost. For this reason, is justified to perform a partial derivatives analysis to reduce the Jacobian construction time.

The Exposed method to calibrate homogeneous matrices represents an alternative to classical methods as DLT and Zhang's methods. It can be used on problems where the position and orientation of two coordinate systems are unknown, and where an intrinsic parameters matrix is not required. A limitation of the proposed methodology is the large dependence of the result on the accuracy in the measurement of the chosen point since all calculations will be based on it.

REFERENCES

[1] Craig J.J. 2009. Introduction to robotics: mechanics and control, 3/E. Pearson Education India.

[2] Choi C. and Christensen H.I. 2012, October. 3D pose estimation of daily objects using an RGB-D camera. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 3342-3349). IEEE.

[3] Wang K., Zhang G. and Bao H. 2014. Robust 3D reconstruction with an RGB-D camera. IEEE Transactions on Image Processing. 23(11): 4893-4906.

[4] Sturm P., Ramalingam S., Tardif J.P., Gasparini S. and Barreto J. 2011. Camera models and fundamental concepts used in geometric computer vision. Foundations and Trends® in Computer Graphics and Vision. 6(1-2): 1-183.

[5] Husty M.L., Pfurner M., Schröcker H.P. and Brunnthaler K. 2007. Algebraic methods in mechanism analysis and synthesis. Robotica. 25(6): 661-675.

[6] Spong M.W. and Vidyasagar M. 2008. Robot dynamics and control. John Wiley & Sons.

[7] Kenwright B. 2012. Inverse kinematics-cyclic coordinate descent (CCD). Journal of Graphics Tools. 16(4): 177-217.

[8] Buss S.R. 2004. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. IEEE Journal of Robotics and Automation. 17(1-19): 16.

[9] Prempraneerach P. and Kulvanit P. 2010. Implementation of Resolved Motion Rate Controller with 5-Axis Robot Manipulator Arm. In The First TSME International Conference on Mechanical Engineering. pp. 1-8.

[10] Di Vito D., Natale C. and Antonelli G. 2017. A comparison of damped least squares algorithms for

VOL. 16, NO. 20, OCTOBER 2021 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2021 Asian Research Publishing Network (ARPN). All rights reserved.

inverse kinematics of robot manipulators. IFAC-PapersOnLine. 50(1): 6869-6874.

[11] Bradski G. and Kaehler A. 2008. Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Inc.

[12] Hartley R. and Zisserman A. 2003. Multiple view geometry in computer vision. Cambridge university press.

[13] Dubrofsky E. 2009. Homography estimation. Diplomová práce. Vancouver: Univerzita Britské Kolumbie.

[14] Přibyl B., Zemčík P. and Čadík M. 2017. Absolute pose estimation from line correspondences using direct linear transformation. Computer Vision and Image Understanding. 161, pp. 130-144.

[15] Zhang Z. 2000. A flexible new technique for camera calibration. IEEE Transactions on pattern analysis and machine intelligence. 22(11): 1330-1334.

[16] Pérez M.A., Cuevas E. and Zaldívar D. 2015. Fundamentos de Robótica y Mecatrónica con MATLAB© y Simulink©. México DF: Afaomega.

[17] Shreiner D., Sellers G., Kessenich J. and Licea-Kane B. 2013. OpenGL programming guide: The Official guide to learning OpenGL, version 4.3. Addison-Wesley.

[18] Förstner W. and Wrobel B.P. 2016. Photogrammetric computer vision. Springer International Publishing Switzerland.

[19] Rohmer E., Singh S.P. and Freese M. 2013, November. V-REP: A versatile and scalable robot simulation framework. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 1321-1326). IEEE.

[20] He K., Gkioxari G., Dollár P. and Girshick R. 2017. Mask r-cnn. In Proceedings of the IEEE international conference on computer vision. pp. 2961-2969.