Colour-Based Object Detection, Inverse Kinematics Algorithms and Pinhole Camera Model for Controlling Robotic Arm Movement System

Mussabayev R. R.
Uniline Group LLP
Almaty, Kazakhstan
ravmus@gmail.com

***Abstract*—Colour-based computer vision algorithm allows to rather precisely distinguish a number of objects disposed on an input camera frame provided that their colours differ from that of a background. Once the border of an object is detected the position of its approximate mass centre may be easily obtained. That coordinate is considered within 2D frame image coming directly from the camera and is subsequently projected to the reference frame associated with the real world by making use of the pinhole camera model and homogeneous coordinates. Given appropriate real world coordinates of the object (which is believed to be put onto a flat plane) iterative Jacobian inverse method provides efficient solution to the problem of inverse kinematics. Thus derived solution is sent to the Arduino microcontroller which in its turn rotates servo joints of the arm by means of the pulse-width modulation (PWM) technique.**

Keywords—computer vision; object detection; inverse kinematics; pinhole camera; robotic arm; Jacobian inverse.

I. INTRODUCTION

A. Preface

Humanity has always been seeking for means of developing intelligent machines for their further employment to a variety of useful applications: starting from homemade hexapod robots to serious industrial welding manipulators and smart humanoid robots based on the use of machine learning. Already achieved tremendous advance in the robotic field of study is obvious to be noticed. Contribution of an intense research into the science of robotics is due to an unstoppable growth of people's interest in the developing of a vast number of robot-like machines of every sort and kind. And, undoubtedly, all efforts will be justified.

The current paper does not claim to be an exhaustive description of all possible and the most optimum algorithms of the robotic arm controlling. Here is shown just one of possible efficient ways and, therefore, there is no frontier to further improvement. The main objective of the current project was to teach robotic arm to perform exact movements sufficient for picking up some arbitrary coloured object from the plane and afterwards put in onto the plate which world coordinates are known beforehand. In the beginning the overall task can appear trivial and inessential but, in fact, without manipulator being able to fulfill such kind of basic actions the more sophisticated applications would have been impracticable.

B. Robotic manipulator

Let us consider the manipulator's structure a little bit closer. The whole system is a sort of rigid multibody and moves by rotating servo motors disposed at its joints. Joints are those very parts of the body where motion occurs. They are connected in the space by stiff links of certain length. The lengths of all links are constants, hence, after being once measured there is no need to do this again.

Actually all joints by their nature are presented by servo motors. Each joint corresponds to one degree of freedom (DOF), and, since the manipulator has 6 servo motors, the system also has 6 degrees of freedom in summary.

General kinds of joints are revolute (rotational) and prismatic (translational) joints [1].

Gripper helps to hold objects and drag them from place to place. In fact gripper converts rotary motion of servo motors into translational movement of the claws. Thus gripper is a translational joint. Remaining joints are revolute and respond to the command by rotation specific joint to the precise angle ranging from 0 to 180 degrees.

Base brings all parts into motion and gives first rough approach to the target object. If all other joints are set to the right angles the gripper will turn out to be at the position suitable for free grasping of the object.

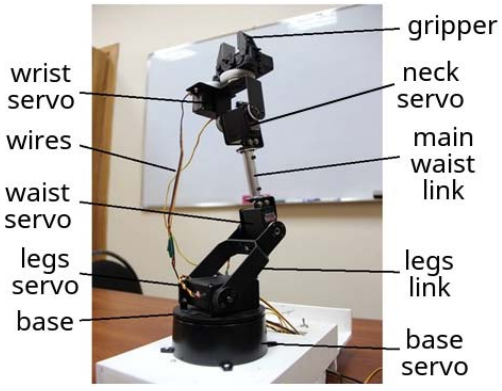

Fig. 1. Main parts of the robotic arm

It is important to point out that in the current project computation of rotation angles for gripper and wrist joints will not be concerned.

C. Goals

The whole objective has been decomposed into several partial tasks which will all be concerned separately:

- Detection of an object on the input camera frame.
- Calculation of the camera's intrinsic/extrinsic parameters and projecting coordinates of the object from the image into the world reference frame.
- Applying Jacobian transpose method for solution of the inverse kinematics problem.
- Developing of the backend software for Arduino microcontroller.

II. Object Detection

One single camera is watching the scene. The camera is constantly turned on and at any moment is able to capture instantaneous image frame from the camera. If that process is repeated over and over again in succession through small intervals of time and every frame is shown on the screen, the result will be exactly the video of the supervised scene.

Whenever target object appears within the field of view of the camera it should be detected, that is bounding rectangle is drawn around an object. To draw a rectangle, the position of the upper left corner and its width and height in pixels are to be known. Now all coordinates are considered within two-dimensional coordinate system of the frame image with the origin associated with the upper left corner with x-axis pointed straight to right and y-axis — downwards.

Let each frame to have 640 on 480 pixels resolution. That means that a frame is represented by the two-dimensional matrix with 3 channels. In other words every entry (pixel) of the matrix with the index (x, y) is a tuple consisting of three values (B, G, R). So our input image initially has the BGR format. B (blue), G (green) and R (red) are figures in range from 0 through 255 which are responsible for the amount of respective colours which being mixed together give certain colour to a pixel. For each pixel in the matrix computer allocates exactly 3 bytes of memory.

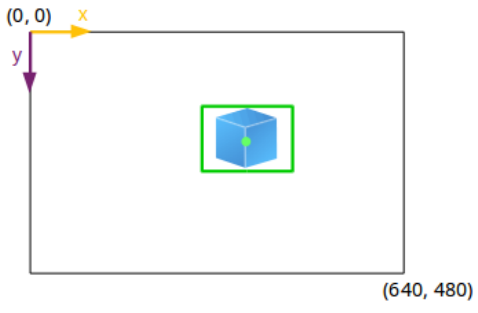

Fig. 2. Input camera frame in the two-dimensional reference system

BGR is the standard colour space but it is not suitable for defining ranges of certain colour. For that purpose first of all BGR is converted to HSV colour space. HSV abbreviation stands for hue, saturation and value. Every colour corresponds to its own particular range in HSV space. Knowing the range for the colour of an object, pixels on the frame with hue pretty close to that of the object are easily noticeable. And then all those pixels are separated from the whole image to get the mask.

What algorithm does is that it performs thresholding of the input frame. The concept of thresholding is straightforward. All pixels lying in between specified colour range are assigned to (255, 255, 255) value and to (0, 0, 0) otherwise. Target object will be marked by white pixels on the black background.

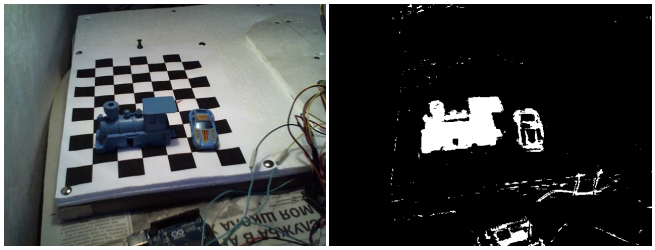

Fig. 3. Initial image (left) and his mask (right) after thresholding

The outline of the object on the mask is indistinct and it is mixed with particles of the scene which also happened to have approximately bluish hue in color. One should notice that the necessary pixels of the object are clustered together whilst other ones are strewn all over the rest part of the mask image.

The next step is to apply border following algorithm to take contours of every cluster made of pixels [2]. Used here algorithm of taking contours is unsusceptible to presence inside clusters a number of little holes. Implemented algorithm as a result returns a sequence of contours of which one with the largest area is to be selected. Utmost points of the contour are chosen as the coordinates for the upper left and lower right corners of the bounding rectangle respectively. The intersection point of diagonals is treated as the tentative centre of mass of the detected object. Thus we are able to determine the precise coordinate of the centre of mass of the object relative to the two-dimensional reference frame of the image.

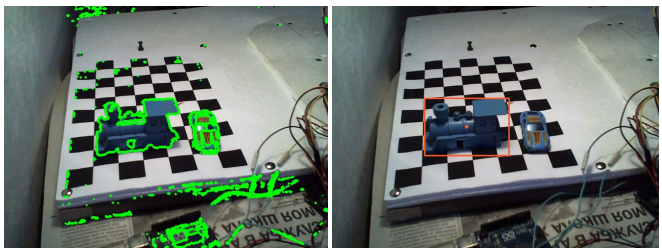

Fig. 4. Image after taking contours (left) and the detected object (right)

III. Projecting Coordinates into World

A. Pinhole camera model

For the most part basic principle of the creation of photo image inside the camera has remained unchanged since the dawn of photography. From the physical perspective light reflecting from an object in the observed scene passes through the frontal aperture disposed at the objective and then hits the image plane located at back of the camera producing

diminished real inverted image of an object. Commonly lens is taken as aperture in cameras. Lens serves to concentrate all beams of light coming from the scene.

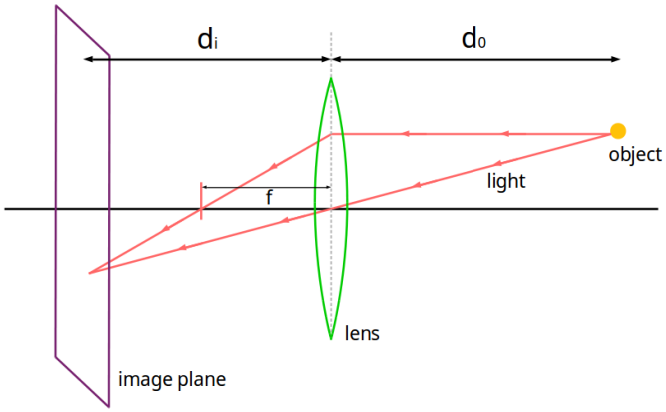

Fig. 5. Usual path of rays through lens in camera

All quantities are related to each other by dint of the so-called thin lens equation:

$$\frac{1}{f} = \frac{1}{d_0} + \frac{1}{d_i} \tag{1}$$

where f is the focal length, d_0 is the distance from an object to the lens and d_i is the distance from the lens to the image plane.

In computer vision it has become customary to employ a simplified in number of ways model of the image capturing. First of all we may entirely neglect the lens effect and consider the camera with lens substituted for infinitesimal aperture, since in theory it does not affect the resulting image. Owing to the fact that in the most cases we have $d_0 \gg d_i$, we might assume that the image plane is placed all the time at focal distance from the optical centre. Also seeing from geometry that the result on the plane is reverse, we can easily change the location of the image plane to be before the optical centre of the system, so that, finally, we obtain the upright image. Undoubtedly, that that model is considerably simplified and evidently is not feasible from the physical point of view, but mathematically it is absolutely identical to the initial one. This whole model is often referred to as the **pinhole** camera model.

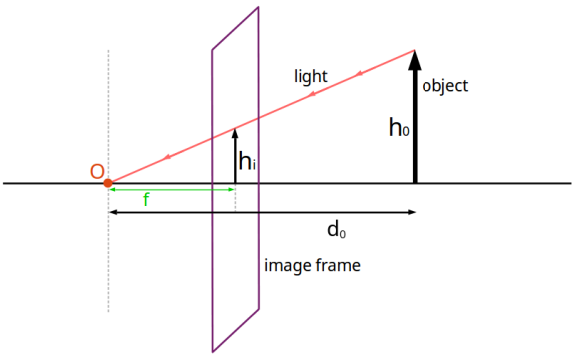

Fig. 6. Simplified pinhole camera model

Guided by the basic rules of making proportions in similar triangles one can deduce the following equation:

$$h_i = f\frac{h_0}{d_0} \tag{2}$$

where h_i is height of the image, d_0 is the distance from the world object to the optical centre, h_0 is height of the world object.

B. Homogeneous coordinates

Homogeneous coordinates are a system of coordinates used in projective geometry, as Cartesian coordinates are used in Euclidian geometry [3]. They bear by far much importance regarding their field of application in computer graphics and computer vision. By the aid of homogeneous coordinates any point, including points on infinity, may be represented by finite coordinates and they allow every projective transformation in projective geometry to be written by a single matrix.

Dimension of the set of homogeneous coordinates is always on one higher than the dimension of the space which is being considered.

To obtain better understanding of homogeneous coordinates let us assume that we have a tuple (X, Y, Z) of homogeneous coordinates representing some point within a plane. Then:

- Multiplying the given triple by a nonzero scalar S, we obtain the triple (SX, SY, SZ) which represents exactly the same point in the plane as the first one.
- In contrast, any two given sets of homogeneous coordinates represent the same point if and only if all their corresponding coordinates are differed on a one and the same scalar factor necessarily not equal to zero.
- To get the Cartesian two-dimensional set of coordinates we simply divide each component of the tuple by its last coordinate. Thus (X/Z, Y/Z) is the coordinate vector of the point in the Euclidian plane.
- To represent the point on infinity we can assign the third component of the triple to be equal to zero.

Important to emphasize that the vector (0, 0, 0) does not actually indicate any point and is hence omitted. Origin in that case is (0, 0, 1).

C. Relation between image plane and world coordinates

Now we are going to delve deeper into the idea of image formation and consider the relationship between what appears on image and where it is located in the world 3D space.

Consider a reference frame whose origin is located at the optical (projection) centre of the system and whose Z-axis (optical axis) passes orthogonally across the image plane and pierces it in some point which is called a **principal point**. It is a three-dimensional reference frame associated with the camera and is called the **standard coordinate system** of the camera.

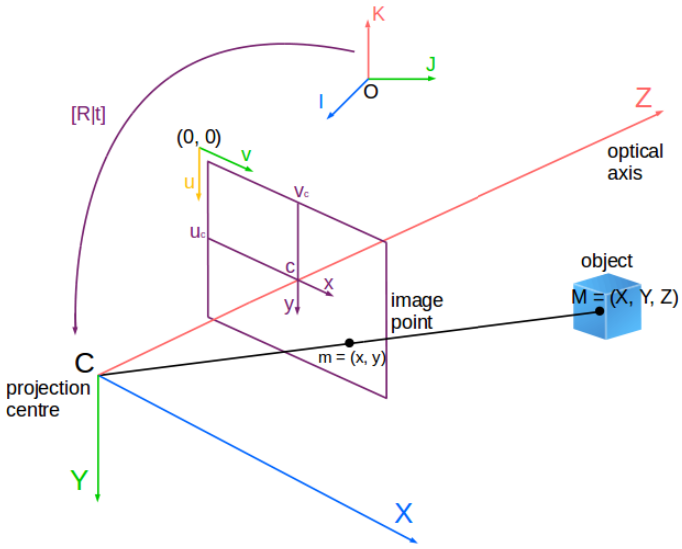

Fig. 7. Relationship between various reference frames of the pinhole camera model

M is some point belonging to the object and having the coordinates (X, Y, Z) respective to the standard coordinate system. The ray of light emitted from it intersects with the image plane leaving the projective point m with the pair of coordinates (x, y). These coordinates are related to the coordinate system originated at the principal point with x and y axes having the same direction as X and Y axes.

Taking proportions in similar triangles as was mentioned earlier, we derive:

$$x = f\frac{X}{Z} \; and \; y = f\frac{Y}{Z} \tag{3}$$

Rewriting the formulas (3) in the form of homogeneous coordinates:

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4}$$

where s is a scalar factor different from zero which indicates that the homogeneous coordinates were used.

Usually inside the image frame upper left corner is taken as the origin and all measures are expressed in pixels. Thus, taking this into account we write down:

$$u = u_0 + \frac{x}{pixel\ width} \; and \; v = v_0 + \frac{y}{pixel\ height} \tag{5}$$

where u_0 and v_0 are coordinates of the principal point c with respect to the upper left corner, and (u, v) is the resulting coordinates of the projection in pixels.

Merging the above formulas (3) and (5) together the following appears:

$$Zu = Zu_0 + f_x X \; and \; Zv = Zv_0 + f_y Y \tag{6}$$

$$f_x = \frac{f}{pixel\ width} \; and \; f_y = \frac{f}{pixel\ height} \tag{7}$$

Or the same (6) in homogeneous representation:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{8}$$

In shorthand notation the general equation (8) appears as follows:

$$sm' = AM' \tag{9}$$

where m' and M' are homogeneous vectors of coordinates of the point in the image and in the standard coordinate system respectively, and the matrix A is referred to as the **perspective projection matrix**. Generally speaking, the camera accomplishes linear projective transformation from the projective space $\mathcal{P}^3$ into the projective plane $\mathcal{P}^2$.

In summary, there are exactly 5 constant parameters: the focal distance f, the coordinates of the principal point u_0 and v_0, width and height of the pixel. These are wholly related to the camera and do not vary from the position and state of the scene being observed. That is the reason for them to be called as the **camera intrinsic parameters**. Similarly, matrix A is frequently called as the **matrix of intrinsic parameters**. It is reasonable to assume that the intrinsic parameters must be preliminarily given by manufacturers and the principal point should be at the precise centre of the image plane, but in practice of computer vision that information is not sufficiently accurate and the principle point may be a number of pixels off from the real centre depending on the quality of manufacture.

Indubitably, (X, Y, Z) coordinates of the object point will not be specified in the form that has been considered so far, i.e. in the standard camera coordinate system. It is just inconvenient to specify them in that way as opposed to some reference system located somewhere in the world and the position of which may be determined beforehand. Such kind of system is represented by IJK with the origin O in the *Fig. 7*.

There exists a marvellous thing in linear algebra that mathematicians extensively employ when they deal with the necessity to convert coordinates of some point considered in one reference frame to the corresponding coordinates within another frame if mutual disposition of the frames may be determined. In other more formal words, there is an isomorphism between two three-dimensional vector spaces and the transformation between them is characterized by the **rotation** matrix R and **translation vector** t. Taking advantage of homogeneous coordinates, these two actions can be combined into the one unique matrix called the **rotation/translation matrix**.

Let us assume that now (X, Y, Z) is believed to be with respect to the world frame. Then in our particular case rotation/translation matrix serves to transform coordinates from the world frame to the standard camera coordinate system:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} and\ t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \quad (10)$$

$$[R|t] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \quad (11)$$

$$M' = [R|t]M \quad (12)$$

And after having rewritten the general equation (8) and (9) we finally derive the overall main formula:

$$sm' = A[R|t]M \quad (13)$$

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (14)$$

The above formula (13) and (14) makes it possible to obtain coordinates of the projection on the 2D image knowing the appropriate coordinates of the point in the world reference system. Additionally, it by far establishes the fundamental relation between the 2D image and the world. But our task requires exactly the opposite.

D. *Inverting the equation*

The first thought that occurs to mind is to invert the equation. However, that turns out to be impossible, since rotation/translation is not square matrix, hence it cannot be inverted. Besides, if we look deeper into the equation and perform all essential matrix multiplication operations we shall inevitably realize that at last we have three unknowns and just two equations. If the vector space of the fundamental system of solutions is derived, it becomes clear that there may be an infinite number of probable solutions.

To proceed in the problem of determining the coordinate vector of the point in 3D space out of its 2D position on the image, at least two cameras watching the scene and the epipolar geometry are needed. But our task includes a useful constraint, namely, the object is always lying on the table, and, having assigned the origin of the world frame somewhere on the surface of the table, we might consider any point to have Z coordinate essentially equal to zero.

Thus:

$$sm' = A\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \quad (15)$$

$$sm' = A\begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \quad (16)$$

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = s\left(A\begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}\right)^{-1} m' \quad (17)$$

$$if\ \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \left(A\begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}\right)^{-1} m', then\ \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} se_1 \\ se_2 \\ se_3 \end{bmatrix} \quad (18)$$

$$s = \frac{1}{e_3};\ X = \frac{e_1}{e_3};\ Y = \frac{e_2}{e_3} \quad (19)$$

E. *Implementation in OpenCV*

For the purpose of getting intrinsic parameters of the camera OpenCV uses the technique of comparing known world coordinates of some well distinguishable objects to their conformable coordinates on the 2D image which are obtained as a result of image recognition process. Using usual chessboard is pretty convenient for that purpose.

The cross-point of the two utmost squares in the upper left corner of the chessboard originates a new world reference frame which is bound to the board itself. Coordinates of all other intersection points are written as (25, 0, 0), (50, 0, 0) and so forth, according to the assumption that the length of the side of each square is equal to 25 millimeters and that the entire measurement is carried out in millimeters as well. Several photos of the chessboard captured from various views are needed for the accurate result.

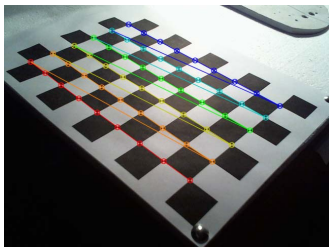

Fig. 8. Recognition of the square corners on the chessboard

And the final result of the foregoing algorithm is shown in the demonstrative look below:

Fig. 9. The final result. X, Y and Z axes are marked by blue, green and red colours respectively

IV. ROBOT ARM KINEMATICS

A. *Base rotation*

Once the object coordinates are reckoned, the angle α is derived from them. α specifies precise angle to which the base joint should rotate in order to set the arm in one plane with the goal object. Thus the problem passes from the 3D space to the 2D plane.

With the height of the base being known the coordinates of the object relative to that reference frame are estimated without difficulty in the plane.

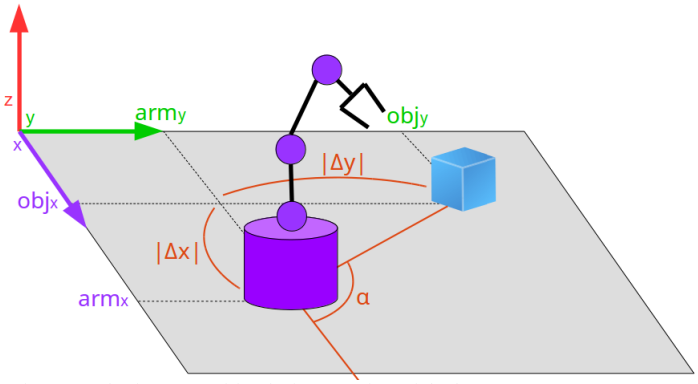

Fig. 10. Robotic arm position before rotation of the base

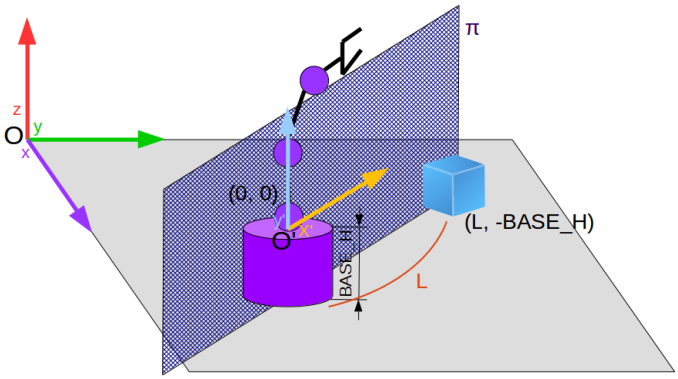

Fig. 11. Robotic arm and the object are situated within one plane π. Now there is a new reference frame O' the origin of which is bound to the first joint

$$\Delta x = obj_x - arm_x \quad (20)$$

$$\Delta y = obj_y - arm_y \quad (21)$$

$$L = \sqrt{\Delta x^2 + \Delta y^2} \quad (22)$$

$$\alpha_d = arccos\frac{\Delta x}{L} rad \quad (23)$$

$$\alpha = \alpha_d \frac{180^o}{\pi rad} \quad (24)$$

α_d is the result angle expressed in radians, and α — in degrees.

Eventually, there remain two-dimensional Cartesian coordinate system, 3 degrees of freedom for remaining joints and all necessary coordinates having been calculated.

B. Forward kinematics

Robot kinematics refers to the methods of obtaining parameters essential to bring kinematic chain to the required position. **Kinematic chain** is an assemblage of stiff links, joints, base and end-effector joined together. Robot kinematics deals with just motion of the system and not at all with the physical aspects of the forces necessary to bring the system into motion which, by the way, relates to the field of study called **robot dynamics**.

Forward kinematics describes the technique of determining the spatial location of the end-effector depending on the known lengths of links and the positions of joints. Actually forward kinematics puts the question: « Given states in which the links and joints stay, what is the proper position of the end-effector for them? ». There are mainly two kinds of solutions to the problem: usage of the geometrical approach and the coordinate transformation method.

Coordinate transformation is rather sophisticated method involving the existence of a coordinate system bound to each joint. Denavit-Hartenberg matrices define relationships between all such coordinate systems. This takes into account possible different sorts of joints (rotational and prismatic) and probable great number of joints and links, and, therefore, is considered as the conventional recommended way to solve the forward kinematics task.

But straightforward application of the geometry would be wise respecting our case since there are just revolute joints forming only 3 degrees of freedom.

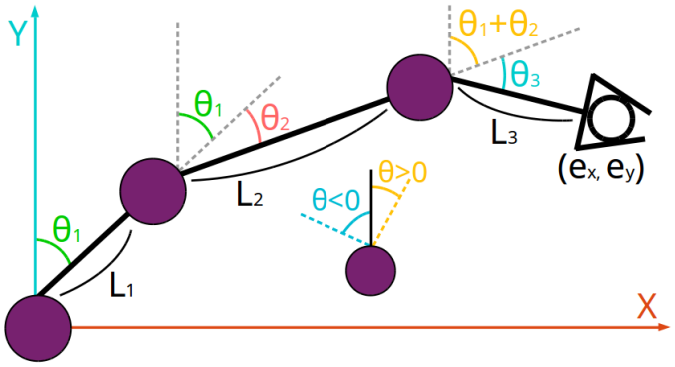

Fig. 12. Geometrical representation of the forward kinematics

$$\theta_i \in [-90°, 90°] \quad (25)$$

$$e_x = L_1 sin\theta_1 + L_2\, sin(\theta_1 + \theta_2) + L_3 sin\,(\theta_1 + \theta_2 + \theta_3) \quad (26)$$

$$e_y = L_1 cos\theta_1 + L_2\, cos(\theta_1 + \theta_2) + L_3 cos\,(\theta_1 + \theta_2 + \theta_3) \quad (27)$$

where $e = \begin{bmatrix} e_x \\ e_y \end{bmatrix}$ is the coordinate vector of the end-effector.

Generally speaking, forward kinematics searches for the vector-valued function whose domain is the vector $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$ consisting of the appropriate angles for all three degrees of freedom of the arm:

$$f(\theta) = e \quad (28)$$

C. Inverse kinematics

Inverse kinematics is a field of robotics whose subject is computing parameters of the joints sufficient to ensure reach of the end-effector to the predefined position. And the problem is considerably hard. It is due to absence of a unique solution. For instance, in our case there is a system of only two equations with three unknown. The problem may have no solutions at all.

According to the fact that solution cannot be derived directly from the forward kinematics equations, the problem solving boils down to apply the iterative methods, capable of providing just one approximate local solution which may be globally inappropriate. Other approaches to the problem of inverse kinematics base on expansion of the forward kinematics equation to the Taylor series and which is usually much easier to be inverted to solve the initial system.

Thus the goal of the inverse kinematics is:

$$\theta = f^{-1}(x) \tag{29}$$

D. Jacobian matrix

Let f be a vector-valued function translating vector coordinates from the arithmetical space $\mathbb{R}^n$ to the arithmetical space $\mathbb{R}^m$. f: $\mathbb{R}^n \to \mathbb{R}^m$. If $x \in \mathbb{R}^n$, then $f(x) \in \mathbb{R}^m$.

The **Jacobian** of the function f is the matrix of all partial first order derivatives of it.

$$J_{mn} = \frac{df}{dx} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \tag{30}$$

$$componentswise: J_{i,j} = \frac{\partial f_i}{\partial x_j} \tag{31}$$

$$where\ 1 \le i \le m\ and\ 1 \le j \le n$$

Setting the stronger condition of the function f to be differentiable at the point x, Jacobian matrix J defines the linear transformation of the vector spaces: $\mathbb{R}^n \to \mathbb{R}^m$, and thus presents the best linear approximation of the function f near the point x. Jacobian matrix generalizes the notion of function derivatives.

Particularly if f is a scalar-valued function, i.e. $m = 1$, then the Jacobian matrix turns into a row J_{1n} — **gradient** of f.

E. Jacobian pseudo-inverse method

In our particular case: $m = 2, n = 3$. f: $\mathbb{R}^3 \to \mathbb{R}^2$. $e = f(\theta) \in \mathbb{R}^2$. $\theta \in \mathbb{R}^3$.

$$J(\theta) = \frac{de}{d\theta} = \begin{bmatrix} \frac{\partial e}{\partial \theta_1} & \frac{\partial e}{\partial \theta_2} & \frac{\partial e}{\partial \theta_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial e_x}{\partial \theta_1} & \frac{\partial e_x}{\partial \theta_2} & \frac{\partial e_x}{\partial \theta_3} \\ \frac{\partial e_y}{\partial \theta_1} & \frac{\partial e_y}{\partial \theta_2} & \frac{\partial e_y}{\partial \theta_3} \end{bmatrix} \tag{32}$$

It is crucial to emphasize that just obtained Jacobian matrix is not constant — it depends on θ:

$$de = J(\theta)d\theta \tag{33}$$

Vector de in the above equation (33) ought to be treated as the difference between vectors which represent the current position of the end-effector and the goal position which has to be reached:

$$de = e_{current} - e_{goal} \tag{34}$$

The overall objective of the algorithm centers around the calculating increment vector $d\theta$ — augmentation to be added to the current θ-vector to take one little baby step towards the goal position. Then the general recurrent formula appears as follows:

$$\theta_{i+1} = \theta_i + d\theta \tag{35}$$

$$d\theta = J^{-1}(\theta)de \tag{36}$$

$$\theta_{i+1} = \theta_i + J^{-1}(\theta)de \tag{37}$$

where θ_i is the domain vector on the current i[th] iteration, θ_{i+1} — vector on the next step and $J^{-1}(\theta)$ is the inverse of Jacobian matrix which needs to be calculated.

Jacobian inverse encloses a little subtlety: it may (will!) not be invertible. Just like in our case, for it is not square matrix. But there exists a way to get through the problem: apply pseudo-inverse technique:

$$de = Jd\theta \tag{35}$$

$$J^T de = J^T J d\theta \tag{36}$$

$$(J^T J)^{-1} J^T de = (J^T J)^{-1}(J^T J)d\theta \tag{37}$$

$$(J^T J)^{-1} J^T de = d\theta \tag{38}$$

$$J^+ = (J^T J)^{-1} J^T \tag{39}$$

where J^+ is the pseudo-inverse of Jacobian.

Using the pseudo-inverse matrix the general equation (37) may be overwritten in the following way:

$$\theta_{i+1} = \theta_i + J^+(\theta)de \tag{40}$$

At last the final issue that remains to be cleared up consists in determining the Jacobian matrix itself according to the equation of forward kinematics. We proceed by straightforward taking all needed partial derivatives separately:

$$J_{11} = \frac{\partial e_x}{\partial \theta_1} = L_1 cos\theta_1 + L_2\, cos(\theta_1 + \theta_2) + L_3 cos\,(\theta_1 + \theta_2 + \theta_3) \tag{41}$$

$$J_{12} = \frac{\partial e_x}{\partial \theta_2} = L_2\, cos(\theta_1 + \theta_2) + L_3 cos\,(\theta_1 + \theta_2 + \theta_3) \tag{42}$$

$$J_{13} = \frac{\partial e_x}{\partial \theta_3} = L_3 cos\,(\theta_1 + \theta_2 + \theta_3) \tag{43}$$

$$J_{21} = \frac{\partial e_y}{\partial \theta_1} = -L_1 sin\theta_1 - L_2\, sin(\theta_1 + \theta_2) - L_3 sin\,(\theta_1 + \theta_2 + \theta_3) \tag{44}$$

$$J_{22} = \frac{\partial e_y}{\partial \theta_2} = -L_2\, sin(\theta_1 + \theta_2) - L_3 sin\,(\theta_1 + \theta_2 + \theta_3) \quad (45)$$

$$J_{23} = \frac{\partial e_y}{\partial \theta_3} = -L_3 sin\,(\theta_1 + \theta_2 + \theta_3) \quad (46)$$

Now we can write down the overall algorithm, which has been described so far, in written form:

- Start in some initial configuration θ_0;
- Define an error metric de (e.g. goal position – current position);
- Compute value of the Jacobian pseudo-inverse with respect to the current θ;
- Estimate increment of θ;
- Iterate until current position of the end-effector is close enough to the goal;

We also might have interpreted Jacobian matrix as a tool that binds together velocities in joints space and velocities in Euclidian space. It can be shown by division both sides of the equation (33) by increment of time:

$$de = J(\theta)d\theta \quad (47)$$

$$\frac{de}{dt} = J(\theta)\frac{d\theta}{dt} \quad (48)$$

$$\dot{e} = J(\theta)\dot{\theta} \quad (49)$$

The essence of Jacobian inverse algorithm is the linear approximation to the actual rotatory movement:

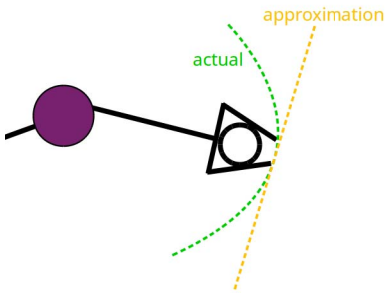

Fig. 13. Jacobian linear approximation

V. Implementation

A. OpenCV

OpenCV (*stands for open source computer vision library*) is a library including a great number of the most robust computer vision, image processing and general-purpose numerical algorithms for solving a vast range of computer vision problems. Majority of the implemented algorithms comes highly optimized. The library has C/C++, Python, Java and other interfaces for the most popular programming languages. OpenCV is licensed under the BSD licence.

For the current project OpenCV helped to get rid of the irrelevant routine like image loading, frame capturing, changing colour spaces and etc.

B. Python

Python is a widely used general-purpose, high-level programming language which popularity is due to the code readability and syntax which allows programmers to express algorithms in few lines of code as opposed to other languages as C++ or Delphi.

Numpy is one of the most powerful build-in Python libraries which had turned out to be useful to facilitate numerical operations and linear algebra in the Python implementation of the project.

C. Arduino

Arduino is an open-source single board microcontroller which software comprises a standard programming language compiler, IDE and a bootloader which loads the compiled code into the chip by means of serial port through USB bus.

Arduino together with PCA9685 board served as an instrument for hooking up and managing a number of servo motors simultaneously.

D. Servo motors

Servo motor provides precise positioning of a joint to which it is stuck on the specified angle. Servo motors do not rotate continually and their rotation is restricted in between fixed angles. Usually they are capable of rotating from 0 to 180 degrees but some kinds of them manage to turn around even to 360 degrees.

Servo motor usually has three wires: 5V+ power, ground and signal cable. Thus the signal can only be either high (5V) or low, and we can change the proportion of time when the signal is high compared to when it is low over a consistent interval of time.

Duty cycle is a metric measured in percentage as a ratio of the interval of time when the signal was high to the period of time through which these signals repeat. The interval when the signal exists is exactly the pulse duration. So the required angle is obtained by servo from the pulse duration. This technique is referred to as a **pulse-width modulation**. Commonly servo waits for a signal every 20 milliseconds what corresponds to the frequency of 50Hz. Pulse duration of 1 ms refers to 0° position, 1.5 ms — 90° (natural position) and 2 ms to 180 degrees of rotation respectively.

VI. Conclusion

The implemented project had done well on main requirements. The arm was able to seize an object at whatever place it took within the arm's workplace provided the camera had it in the field of vision.

Fig. 14. Robotic arm in the intial position (left) and gripping the object (right)

Fig. 15. Robotic arm holding (left) and releasing the object (right)

If a single camera does not suffice to cover the entire scene, two or more cameras viewing the scene from various perspectives would work perfectly.

In spite of the apparent flawless working, the proposed method is unfit to more complex robotic systems which involve other types of joints (e.g. prismatic). Furthermore, it suffers from the lack of performance that is due to the necessity to manually pick an initial setting of the parameter vector.

The project's peculiarity consists in that the research has started quite recently and there remains a multitude of techniques and unrealized ideas that could be appropriate to rectify the existing defects as well as to ameliorate the overall robustness and extend the application domain of the current state of the project.

Object detection technique that has been considered so far generally does not take into account specific forms and is unable to distinguish variegated objects. Machine learning neural network approach to the problem is going to be used so that efficiency of the object detection would increase appreciably.

Moreover, the constraint for an object to have Z coordinate essentially equal to zero will be removed, thereby by the use of epipolar geometry with stereopair coming simultaneously from two cameras the program will be able to evaluate its full location in space, thus enabling to operate with objects irrespective of any constraint.

The inverse kinematics algorithm will be improved in order to accommodate it to various types of kinematic systems and to speed up the computation. The future version will also try to choose the optimal paths involving the least energy and time possible to reach the goal.

VII. References

[1] Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods, Samuel R. Buss, University of California, San Diego, October 7, 2009;

[2] Topological Structural Analysis of Digitized Binary Images by Border Following, Satoshi Suzuki, Graduate School of Electronic Science and Technology, Shizuoka University, December 16, 1983;

[3] Wikipedia — The Free Encyclopedia: https://en.wikipedia.org/wiki/Homogeneous_coordinates;