International Journal of Engineering & Technology
CENTRE OF PROFESSIONAL RESEARCH PUBLICATIONS
ISSN 2049-3444

International Journal of Engineering and Technology Volume 2 No. 10, October, 2012

An Inverse Kinematic Analysis of a Robotic Sealer

Akinola A. Adeniyi [1], Abubakar Mohammed [2], Aladeniyi Kehinde [3]
[1]Department of Mechanical Engineering, University of Ilorin, Ilorin, Nigeria
[2]Department of Mechanical Engineering, Federal University of Technology, Minna, Nigeria
[3]Department of Science Laboratory Technology, Rufus Giwa Polytechnic, Owo, Nigeria

ABSTRACT

A planar robotic sealing or brand stamping machine is presented for an automated factory line. The appropriate time to seal or to stamp an object is basically determined by a motor controller which relies critically on whether or not the object is in the best position. The extent of protraction and retraction of the piston head is largely dictated by an infrared sensor. Given the extent to protract or retract the piston head, the angular displacements of the link required are determined using the Inverse Kinematic (IK) techniques. The inertia and gravity effects of the links have been ignored to reduce the complexity of the equations and to demonstrate the technique.

Keywords: *Forward Kinematics, Inverse Kinematics, Robotics, Sealer.*

1. INTRODUCTION

An automated factory uses a number of mechanical links electronically controlled to achieve tasks. The benefits of factory automation are many and of strategic importance to management [1]. Standard mechanical links are usually powered with electrical motors, pneumatic systems or solenoids. In a manually operated machine, the human performs visual checks and other standard checks that are to be replicated by automation. The interest of this work is centered on a hypothetical sealing machine which is used for stamping some signatures and logos as done in a branding factory line. Inverse kinematic analysis is applied to enable us determine angular displacements of the link. Kinematics involves the study of motion without consideration for the actuating forces. Inverse Kinematics (IK) is a method for determining the joint angles and desired position of the end-effectors given a desired goal to reach by the end effectors [1].

A feasibility of using a PID controller was studied by *Nagchaudhuri* [2] for a slider crank mechanism but without an offset. *Tolani et al* [3] reviewed and grouped the techniques of solving inverse kinematics problems into seven. The techniques are the Newton-Raphson's method and its other variants. There are the Jacobian and the variants with pseudo-inverse (otherwise known as the *Moore-Penrose inverse*) for square or non-square Jacobian. Other methods are the control-theory based and the optimisation techniques. A number of authors [1, 4-7] have proposed algorithms for solving IK problems which include but not limited to Neural Network algorithm, Cyclic Coordinate Descent closure and Inexact strategy, but like every other techniques for a given problem the choice of method depends on the specifics of the problem. Buss [8] discussed the Jacobian transpose, the *Moore-Penrose* and the Damped Least Squares techniques. In terms of computational cost, the Jacobian transpose method is the cheap but can perform poorly based on the robot configurations. In this work the Jacobian transpose technique ill-performed but the Jacobian Inverse technique is suitable and more so it is a simple 2D planar representation of the problem with only 4 degrees of freedom.

2. OPERATIONS OF THE ROBOTIC LINK

Fig. 1 shows the schematic diagram of the robotic sealing system. The capping or stamping is achieved with the piston or ram head, **P**. **C** is the conveyor line. The caps or the branding heads are placed in position and sensed by an infrared sensor, **S**. The instruction to seal or brand is dependent on feedback from the sensor. If the item to be branded, capped or stamped is out of place at the instance when the ram head was going to touch, the sensor feedback will be to retract the head. It can also be to not go too far. There can be a range of feedback to the motor controller, **M**. This kind of control system is similar to what a human operator would do if it were manually operated. The use of sensors and fast responding motor controller will make this hypothetical machine a very useful tool in a factory performing this kind of mundane task. This factory sub-line is a simple slider-crank mechanism with actuator arm **A**.

In clearer terms, the instructions would be to *press the piston* ram to seal if the cap and the container are in line; to *reverse* the piston in case of a jam; to *not press* the piston ram if either the container or the cap is absent; to

press further if the seal length is shorter than expected as may be caused by wear and tear. This clearly shows that the piston determines the angle of the link or the direction or action of the motor. This is an inverse kinematics problem. The sensor feedback part is much of a control engineering problem, not considered in this paper.

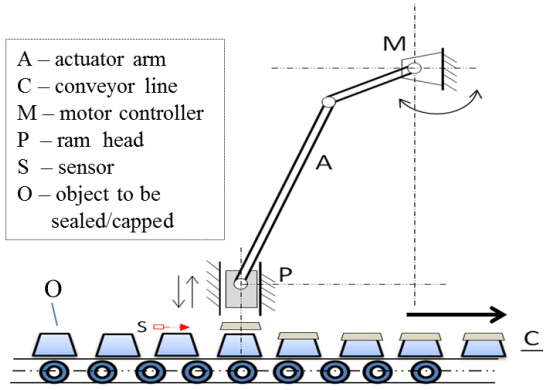

Fig. 1: The robotic sealing rig schematic

3. ANALYSIS

Fig. 2 is a representation of the slider-crank mechanism. There is an offset, *f*, of the piston axis from the motor axis, O_1. O_2 is the axis of the piston with moving coordinates (*x*,*y*). The motor rotates clockwise or counter clockwise about O_1. If the crank makes displacement Δs on the piston plane, it is equivalent to a motion of Δe_x and Δe_y. This motion is caused by the crank making an angular motion clockwise or counter-clockwise, $\Delta\alpha$. The angle between the connecting rod and crank makes an angular displacement of, $\Delta\beta$. This also means the angular shift of $\Delta\lambda$ is made between the connecting rod and the piston or ram plane.

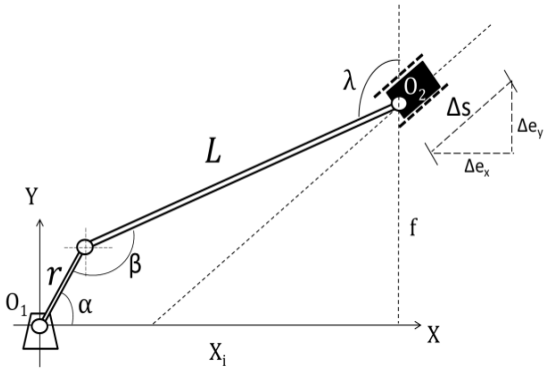

Fig. 2: The offset slider crank (*Cartesian coordinate world*)

In a computer game application for these, the angles would be explicitly required so that the links do not "physically disjoint"; for a physically connected link, the motor controller only would need the instruction to move only the crank.

3.1 The World

Cartesian coordinate system is adopted. Clockwise is positive and motion to right and upwards are positive. The Top Dead Centre (TDC) is attained when the crank, radius r, and the connecting rod, length *l*, are in line. This is attained when $\alpha = \alpha_{TDC} = Sin^{-1}(f_m/(r+L))$. f_m is the maximum variable offset based on the geometry. The Bottom Dead Centre (BDC) is reached when $\alpha = \alpha_{BDC} = \alpha_{TDC} + \pi$. The TDC and BDC with the variable offset are shown in **Fig. 3**.

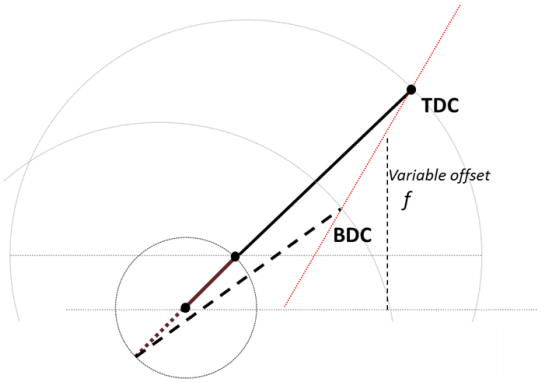

Fig. 3: The Top and Bottom Dead centre

The piston has been constrained to move only in planar direction, on the vector of $\overrightarrow{\Delta s}$. In this work, the direction vector is $\widehat{\Delta s} = \hat{\imath} + \hat{\jmath}$, making the plane at 45° to the horizontal.

3.2 The Forward Kinematics

The displacement caused by the motor moving clockwise from the position in **Fig. 2** is represented in equation (1). Where subscripts *(i,f)* are respectively mean initial and final values. The position at f is reached in reality smoothly for a rotating crank, but the smoothness can be reached in fine incremental steps, in the numerical approach. At the end of the stepped increments, the final displacement to the goal is seen as a function of angular parameters given as:

$$\Delta x = X_f - X_i = x(\alpha_f, \beta_f, \lambda_f) - x(\alpha_i, \beta_i, \lambda_i) \quad (1)$$

The linear dependence of the angles, in this problem, can help to reduce the number of degrees of freedom to compute in equation (1). It can be shown that $\lambda = \beta + \alpha - \frac{\pi}{2}$, thereby making $X_i = X_i(\alpha, \beta)$. Using trigonometry, the instantaneous initial, arbitrary, position of the piston in **Fig. 2** is given by Equation (2).

$$X_i = rCos(\alpha) - LCos(\alpha + \beta) \quad (2)$$

$$Y_i = rSin(\alpha) - LSin(\alpha + \beta) \quad (3)$$

The Jacobian matrix for $X_i(\alpha, \beta)$is given in equation (4) and simplified to equation (5).

$$J=\begin{bmatrix}\frac{\partial X_i}{\partial \alpha} & \frac{\partial X_i}{\partial \beta}\\ \frac{\partial Y_i}{\partial \alpha} & \frac{\partial Y_i}{\partial \beta}\end{bmatrix} \quad (4)$$

$$J=\begin{bmatrix}-rSin(\alpha) & LSin(\alpha+\beta)\\ rCos(\alpha) & -LCos(\alpha+\beta)\end{bmatrix} \quad (5)$$

Computing the new piston position involves solving equation (1). The new coordinate of the piston by the first term of expansion of the Taylor series can be shown to be given in equation (6). θ is the vector of the robot angular displacements for the related links. Mathematically, $\theta = [\theta_1\ \theta_2\ \ldots..\theta_i\ ]^T$. Here, we have $\theta = [\ \alpha \quad \beta \quad \lambda\ ]^T$. Therefore the current position of the piston or the pressing head is approximately given in equation (6). It should be noted that β can be measured from the horizontal to further reduce the equation sets, this is referred to as β_o elsewhere in this paper.

$$X_f = X_i + J\Delta\theta \quad (6)$$

3.3 Inverse Kinematics

The problem is not that of solving for X_f given X_i and θ but it is that of solving for θ given X_i, and the desired X_f. This is iteratively implemented such that the target displacement of the piston is given as $\Delta x = \sum_i S_i$. This is a vector of the piston displacement and can be represented as $\Delta x = [\Delta S_x \quad \Delta S_y \quad \Delta S_z]$. Since this is a planar problem with no displacements in the other directions, it reduces to a $\Delta x = \Delta S_x$. To smoothen the possible jerk or jumpy effect, this can be stepped using a factor of μ which can be selected intuitively based on the ratio of r to L but $\mu \leq 1$ and $[J]^{inv}$ is the inverse of Jacobian matrix. The algorithm checks if the target has been reached or not. Iteration is stopped when the solution is within a pre-determined level of error or a maximum number of iterations. The choice of these limiting values should depend on the response time acceptable. This can be critical for a real time application.

$$\Delta\theta = \mu[J]^{inv}[\Delta x] \quad (7)$$

4. RESULT AND DISCUSSIONS

Consider a current orientation of the robotic arm at any arbitrary position with the piston head at a position P^1. Suppose the sensor system requires the piston to move to a target new position P^2. The simulation is done for several arbitrary starting positions of the crank and results are similar for reachable targets. Supposing the crank angle is at a current orientation with crank angle of -5°, and there is an instruction from the sensor to retract the piston ram head by 0.1 times the crank arm length. The simulation instructs the crank proceeds to counter clockwise by 15.58°, this corresponds to an increase of β_o to 19.26° and correspondingly, λ reduces to 86.32°. Fig. 4 shows the simulation progress of the piston head from a current position P^1 to the new target P^2 and the number of iterations done.

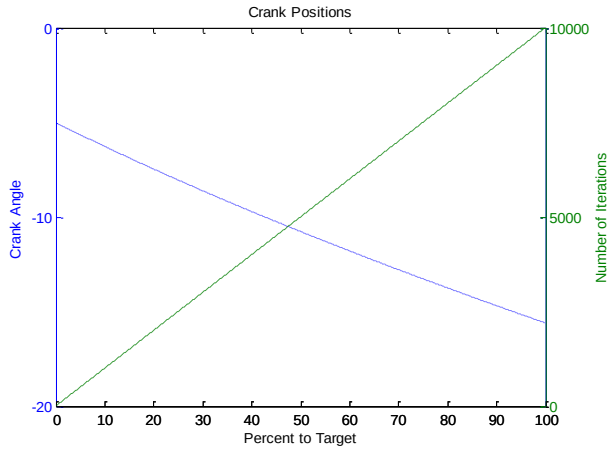

Fig. 4: Crank Position and Iteration with the Jacobian Inverse Matrix

The technique used is the Jacobian inverse technique. The Jacobian transpose technique is not predictable for the same problem and in this case, the solution settles to a local minimum for only one of the angles but the convergence rate is faster, see Fig. 5.

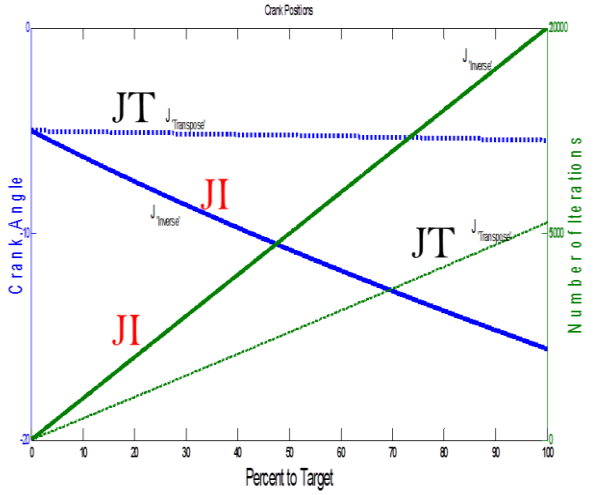

Fig. 5: Crank Positions using the Inverse and Transpose of the Jacobian Matrix

If there is a request to a physically unreachable target, such as to a more than the TDC or BDC locations, P^3, the simulation runs and stops after the maximum number of iterations or if the Jacobian Matrix becomes un-invertible, Fig. 6.

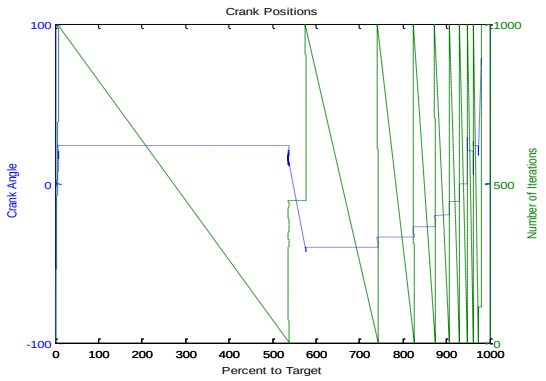

Fig. 6: Unreachable Target situation

5. CONCLUSION

This paper is focused on the application of the Inverse Kinematics technique to the analysis of a robotic link, such as obtained in a sealer of an automated factory, without consideration for the effects of inertia effects. The Jacobian inverse technique, as mentioned in literatures, is more reliable in this application. The Jacobian transpose approach is not reliable. This paper has demonstrated the application of the inverse kinematics to a simple robotic sealer; the piston is instructed to retract by 0.1 units as a test case. The new crank angle was found more accurately with the Jacobian Inverse technique better that the Jacobian Transpose technique. The problem can be extended to include the dynamics for possible selection of the optimal driving torque or electric motor selection for the driving parts.

REFERENCES

[1] S. Tejomurtula and S. Kak, "Inverse Kinematics in robotics using neural networks," *Information Sciences*, vol. 116, pp. 147-164, 1999.

[2] A. Nagchaudhuri, "Mechantronic Redesign of Slider Crank Mechanism," in *ASME International Mechnical Engineering Congress & Exposition: IMECE2002*, New Orleans, Louisiana, 2002.

[3] D. Tolani, A. Goswami, and N. I. Badler, "Real-Time Inverse Kinematics Techniques for Anthromorphic Limbs," *Graphical Models*, vol. 62, pp. 353-388, 2000.

[4] S. K. Saha and W. O. Schiehlen, "Recursive Kinematics and Dynamics for Parallel Structured Closed-Loop multibody Systems," *Mechanics of Structures and Machines*, vol. 29, pp. 143-175, 2007.

[5] X. Wang, "A behavior-based inverse kinematics algorithm to predict arm prehension postures for computer-aided ergonomic evaluation," *Journal of Biomechanics*, vol. 32, pp. 453-460, 1999.

[6] A. C. Nearchou, "Solving the inverse kinematics problem of redundant robots operating in complex environments via a modified genetic algorithm," *Mechanism and Machine Theory*, vol. 33, pp. 273-292, 1998.

[7] M. J. D. Powell, "Some Global Convergence Properties of a variable metric Algorithm for Minimization without Exact line searches," in *Symposium in Applied Mathematics of the American Mathematical Society and the Society for Industrial and Applied Mathematics*, New York City, 1976.

[8] S. R. Buss, "Introduction to Inverse Kinematics with Jocobian Transpose, Pseudoinverse and Damped Least Square methods," University of California, San Diego2009.

Appendix

Source Code

```
%%MATLAB/Octave Script
%%Solver for the IK implementation of the offset slider crank
%% Initialising
%%Crank Radius R, Con-Rod Length L
%% World Starting Geometry
%%==============INITIALISE====================
R=1; L=7*R; alpha=-5; beta=18.3;  PlaneInc=45;
PercentError=0.1; MaxITER=10000;
%%============================================
Pie=3.1415926535897932385;
toRad = inline('x.*3.1415926535897932385/180');
toDeg = inline('x.*180/3.1415926535897932385');
%%The Piston is to Move by delta_S_Goal from a current position
alpha=toRad(alpha);  beta=toRad(beta); PlaneInc=toRad(PlaneInc);
Theta=[alpha beta (beta+Pie/2)];
a=alpha;b=beta;
%% CHANGE Here        %% This for the Goal -?? < KK < ??
%% KK is a Factor of the Crank Radius
   KK=-0.1;
delta_S_Goal=KK*R;
 delta_S=0.;   Rho=1;
PercentClose=[];   CrankAng=[]; Iterations=[];
if abs(delta_S_Goal)>0.1
 %% eg avoid too big step
 Rho=10;
 end;
 delta_S_Goal_i=0;
   Xi=R*cos(alpha)+L*cos(beta);         Yi=R*sin(alpha)+L*sin(beta);
   Xi_i=0;    Yi_i=0;deltaTheta=[0 0 0];              ds=delta_S_Goal/Rho;
for ITRSteps=1:Rho
delta_S_Goal_i =delta_S_Goal_i+ds;
MaxIterations=0;   reached=false;
Xi_i=Xi_i+Xi;          Yi_i=Yi_i+Yi;       Xpos=[Xpos Xi_i];
deltaX=ds*[cos(PlaneInc) sin(PlaneInc)];
while(reached==false && MaxIterations<MaxITER)
  MaxIterations=MaxIterations+1;
%% J =[a b; c d]                        %% JI=Jacobian Matrix inverse, JT=Jacobian Transpose
   a=-R*sin(alpha);b=-L*sin(beta);c=R*cos(alpha);
  d=L*cos(beta);detm=a*d-b*c;
   J=[a b;c d];
%% Avoid Division by zero NO Link Movements
if (detm~=0)
  detm=1/detm;
else
 disp('Division by Zero -!!! Impossible Link Orientation')
end;
JI=detm.*[d -b; -c a];  %% JI=inv(J)
%% detm=1; %% JT=[a b; c d]';
%% delta_S_Goal_i=Piston Displacement size
%%deltaTheta=JI*deltaX';
deltaTheta=0.0001*JI*deltaX' ;
alpha =alpha + deltaTheta(1);
beta  =beta  + deltaTheta(2);
%%Check if goal is reached
Xf=R*cos(alpha)+L*cos(beta);
Yf=R*sin(alpha)+L*sin(beta);
delta_Sf=(Xf-Xi_i)/cos(PlaneInc);
%%-------------------------------------------------
reached=abs(100*(delta_Sf/delta_S_Goal_i));
PercentClose=[PercentClose reached];
CrankAng=[CrankAng toDeg(alpha)];
Iterations=[Iterations  MaxIterations];
  if (reached>=(100-PercentError) && reached<=(100+PercentError))
```

```
   reached=true;
  else
   reached=false;
  end
end
end
%%--OUTPUT
Alpha_Beta_Lamda=toDeg([alpha beta (alpha+beta-Pie./2)])
       delta_Sf=[(Xf-Xi)/cos(PlaneInc) (Yf-Yi)/sin(PlaneInc) delta_S_Goal]'
        MaxIterations
[x,y_axis1,yaxis2]=plotyy(PercentClose,CrankAng,PercentClose,Iterations,'plot');
set(get(x(1),'Ylabel'),'String','Crank Angle')
set(get(x(2),'Ylabel'),'String','Number of Iterations')
xlabel('Percent to Target')                    title('Crank Positions')          set(y_axis1,'LineStyle',':')              set(yaxis2,'LineStyle','--')
```