Metodología para encontrar Matrices de Transformación Homogénea para la Cinemática de Mecanismos con Matlab y VRML

Methodology to find Homogeneous Tranformation Matrices for Kinematics of Mechanisms with Matlab and VRML

Jesús Aureliano Esquivel Cárdenas[1], Javier Ruiz Tello[1], Gerardo Rincón Maltos[2]

[(1)]Docente ([(2)]Estudiante) de la Maestría en Ingeniería Eléctrica con acentuación en Control Automático de la Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autónoma Coahuila. Monclova, Coahuila, México.
email:jesus.esquivel@uadec.edu.mx

Recibido 10/09/13, aceptado 27/12/2013

RESUMEN

Este trabajo resuelve el problema de encontrar las matrices de transformación homogénea, [1]-[7], en el estudio de mecanismos, especialmente la robótica, de diseños 3D previamente elaborados o exportados a archivos WRL, [8]-[13] en los cuales se desconocen las medidas de las dimensiones de posición y orientación de los pivotes de accionamiento. La metodología es gráfica y desarrollada mediante la programación de interfaces en Matlab, [14]-[18], a mundos virtuales programados en VRML. Conceptos como los parámetros Denavit-Hartenberg los cuales son tres y la variable de unión son importantes para minimizar la dependencia paramétrica entre los sistemas de coordenadas los cuales se logra siempre y cuando estos satisfagan una serie de condiciones las cuales son difíciles de encontrar en diseños 3D carentes de información o entre los cuales las dependencia de Transformaciones los cuales son nodos nativos de la programación VRML satisfacen relaciones anidadas y complejas.

Palabras clave: Realidad Virtual, VRML, Matlab, GUI, Denavit-Hartenberg,

ABSTRACT

This work solves the problem of finding the matrix of homogenous transformation, [1]-[7], in the study of mechanisms, in special Robotics, of 3D designs previously developed or exported to WRL files, [8]-[13] where are position and orientation measures of the actuating pivot are missing. The approach taken is graphical and is implemented relying in special functions of Matlab, [14]-[18], for making interaction with VRML worlds. Concepts like parameters Denavit-Hartenberg which are four, are important due they are the minimal quantity of parameters necessary for relating two coordinate frames, are hard to find whenever there are no data of dimensions or size relating position and orientation between different objects or when the Transformations, native nodes of the VRML, have nested, hence complex, structures.

Key words: Virtual Reality, VRML, Matlab, GUI, Denavit-Hartenberg,

1. INTRODUCCIÓN

El diseño de mundos virtuales, [19]-[22], se desarrolla mediante herramientas o software diversos tanto comercial, como el *3D Studio* de Autodesk, o el *Blender* que es de acceso libre. El problema que se resuelve en este trabajo es el de ubicar mediante una forma gráfica la posición de los diferentes sistemas de coordenadas involucradas en los movimientos de un robot virtual cuya información se desconoce, así como los parámetros Denavit-Hartenberg (DH) que logran la relación entre ellos. Existen herramientas para el diseño de archivos WRL los cuales codifican el lenguaje VRML, *e.g.* el *vrbuilder2*, *el Flux Studio*, etc. que permiten desarrollar mundos virtuales con geometrías muy modestas. La metodología propuesta en este estudio inicia desde que se desarrollan los objetos programados directamente en VRML utilizando términos propios de este software hasta que su manipulación con una Interface Gráfica de Usuario (GUI - por sus siglas en Inglés) del Matlab para resolver el problema de la Cinemática Directa e Inversa.

2. MÉTODO GRÁFICO - PROGRAMACIÓN

En esta sección estudiaremos los conceptos principales de nuestro método exponiendo el diseño de un manipulador de dos grados de libertad en un plano. Para ello escribimos un archivo WRL, programado en VRML, y particionado para mostrarlo en las siguientes tres listas. En la Lista 1 se muestra el punto de visión de una cámara para visualizar el manipulador centrado en la escena.

Lista 1. Código **Viewpoint**
List 1. Viewpoint code

```
#VRML V2.0 utf8
DEF Enfrente Viewpoint {
fieldOfView 0.785398
orientation 0 -1 0  0.286
position    0.295417 1.6 24.244
}
```

Lista 2. Código **Link1**
List 2. Link1 code

```
DEF Link1 Transform {
    rotation    0 0 1 0
    children [
        USE VinculoLink2
        Transform {
            translation 0 0 0
            rotation    1 0 0 1.5708
            children Shape {
            appearance  Appearance {
              material     Material {
              diffuseColor 0.28 0.29 0.8
                    } }
            geometry    Cylinder {
                } } }
         Transform {
            translation 5 0 0
            children Transform {
                translation -4.76 3.2 0
                children Shape {
                    appearance Appearance {
                       material Material {
                 diffuseColor 0.76 0.24 0.32
                        } }
                    geometry Box {
                        size 10.5 1.7 0.6
                    } } } } ]}
```

Lista 3. Código **Link2**
List 3. Link2 code

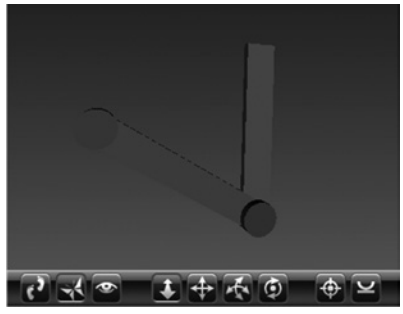
```
DEF VinculoLink2 Transform {
    children Transform {
        translation 5.3 0 0
        children DEF Link2 Transform {
            rotation  0 0 1 2.10487
            children [
    Transform {
        translation 0 0 0
        rotation    1 0 0  1.5708
        children Shape {
         appearance  Appearance {
           material   Material {
         diffuseColor 0.225 0.27 0.8
             } }
         geometry    Cylinder {
          } } }
    Transform {
        translation 5 0 0
        children Transform {
            translation 0 0 0
            children Shape {
  appearance Appearance {
    material  Material {
    diffuseColor 0.8 0.29 0.225
                    } }
   geometry    Box {
        size    10.5 1.7 0.6
   } } } } ] } } }
size    10.5 1.7 0.6
} } } } ] } } }
```

El código de la Lista 2 se ilustra el Eslabón 1 (Link1) el cual consiste simplemente en un cilindro y una caja rectangular los cuales son nodos Transform modificados para acoplarse a la orientación y posición adecuada y darle forma al eslabón cuyo campo de manipulación es rotation el cual se encuentra adaptado para mover el eslabón sobre el eje z. Los ejes x y y se ajustan de tal forma que se reúnan las condiciones del método DH. La selección exacta la haremos mediante una GUI del Matlab que manipule a los cuatro parámetros de tal forma que el marco de coordenadas satisfaga en forma inherente las condiciones DH.

El código de la Lista 3 define el nodo VinculoLink2 para utilizarlo en la Figura2. El Link2 está inmerso dentro del código debido a que es necesario modificarlo en su posición y orientación en el espacio antes de utilizarlo directamente mediante el campo orientation en el eje z.

En la Figura 1 se muestra el manipulador generado por este programa y visualizado mediante el software *Cortona3D*.

Figura 1. Brazo de dos Grados de Libertad
Figure 1. Two DOF Arm



2.1 Método gráfico para generar la matriz de transformación

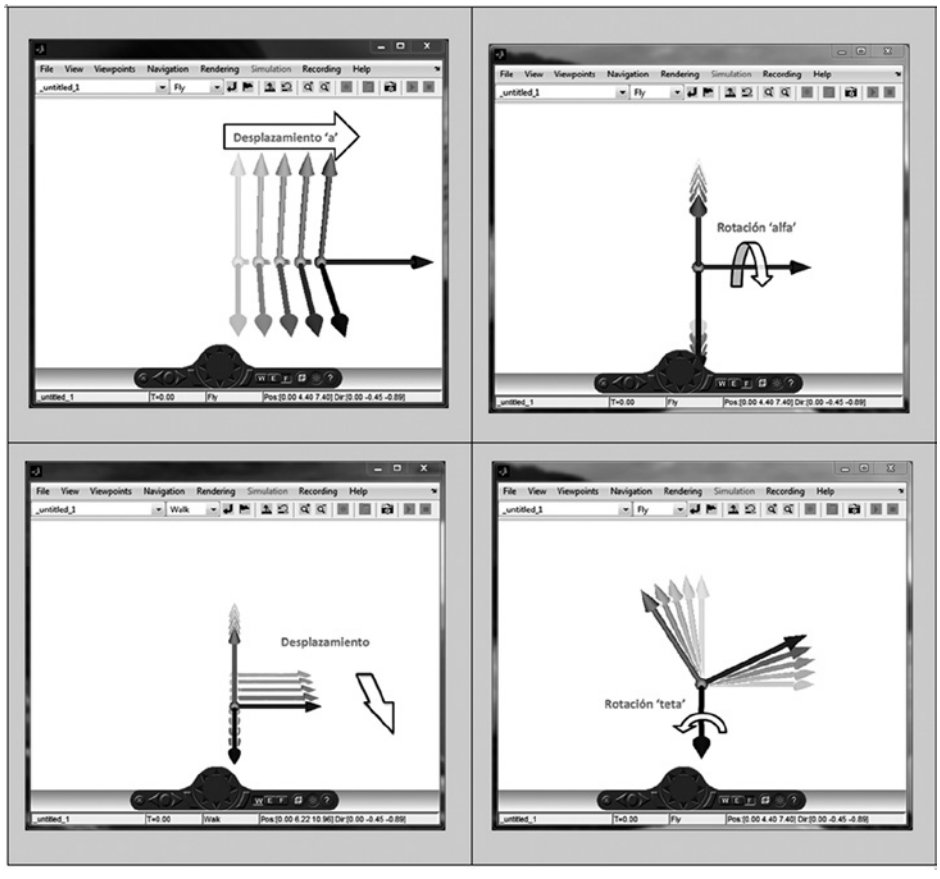
La matriz de transformación homogénea relaciona dos sistemas de coordenadas y se define mediante la transformación canónica establecida por los parámetros DH,

$$A(\theta,d,a,\alpha)=\begin{bmatrix}c_\theta & -s_\theta & 0 & 0\\ s_\theta & c_\theta & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & d\\ 0 & 0 & 0 & 1\end{bmatrix}$$
$$\times\begin{bmatrix}1 & 0 & 0 & a\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & c_\alpha & -s_\alpha & 0\\ 0 & s_\alpha & c_\alpha & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

La rotación sobre el eje x actual se denota por θ, posteriormente la segunda transformación es un desplazamiento d sobre el eje z. La siguiente transformación es un desplazamiento sobre el eje de las x actual, y finalmente una rotación sobre este mismo eje pero del sistema actual. Es importante recordar que los cuatro parámetros definen un rango mínimo de transformaciones que relacionan dos sistemas de coordenadas tridimensionales. Es decir, no es posible obtener la relación con estos parámetros, en general, de cualquier par de sistemas de coordenadas con posiciones y orientaciones aleatorias. El número de parámetros debe incrementarse de acuerdo a las necesidades, sin embargo no puede rebasar los doce lo cual es el número de entradas de la matriz de transformación de los primeros tres renglones. En su versión intermedia solamente seis, tres para orientación y tres para traslación, y en su versión mínima cuatro, que en nuestro caso los derivamos de los criterios establecidos por Denavit y Hartenberg.

Lo inicializaremos en cero para tener empalmados los sistemas de coordenadas. Al variar los parámetros mantenemos solo se mueve uno de ellos. La idea es utilizar esta técnica para buscar emparejar visualmente este sistema variable con un tercero que en su momento se encontrará fijo en un eslabón de robot para derivar de esta manera la matriz de transformación homogénea. Es claro la imprecisión de los cálculos involucrados y el riesgo de implementarlos en un diseño mecánico real. Sin embargo no es el tipo de problemas considerados en este trabajo. Los cambios en cada uno de los parámetros se muestran en las Figura 2.

Figura 2. Cambio del valor de los parámetros
Figure 2. Change of parameters' value



Los cambios en la rotación y traslación de los nodos definidos por **Transform** se implementaron mediante los siguientes fragmentos de código de Matlab codificados en una GUI mediante la herramienta GUIDE,

Listas del Programa GUI de Matlab

Lista 4. Parámetro a
List 4. Parameter a

```
global T;
a = get(hObject,'value');
T(:,:,3) = [1 0 0 a;0 1 0 0;...
    0 0 1 0;0 0 0 1];
actualizar();
```

Lista 5. Parámetro d
List 5. Parameter d

```
global T;
d = get(hObject,'value');
T(:,:,2) = [1 0 0 0;0 1 0 0;...
    0 0 1 d;0 0 0 1];
actualizar();
```

Lista 6. Parámetro a (alfa)
List 6. Parameter a (alpha)

```
global T;
a = get(hObject,'value');
T(:,:,4) = [1 0 0 0;0 cos(a) -sin(a) 0;...
    0 sin(a) cos(a) 0;0 0 0 1];
actualizar();
```

Lista 7. Parámetro th (teta)
List 7. Parameter th (theta)

```
global T;
th = get(hObject,'value');
T(:,:,1) = [cos(th) -sin(th) 0 0;sin(th) cos(th)
0 0;...
    0 0 1 0;0 0 0 1];
actualizar();
```

Lista 8. Función Actualizar
List 8. Function Actualizar

```
function actualizar()
global T origen;
A = eye(4);
for i=1:4
    A = A*T(:,:,i);
end
R = A(1:3,1:3);
th =  acos((trace(R)-1)/2);
k =[R(3,2)-R(2,3) R(1,3)-R(3,1) R(2,1)-R(1,2)]/
(2*sin(th));

origen.translation = A(1:3,4)';
origen.rotation = [k th];
```

La transformación homogénea en la Lista 8 tiene la forma

$$A = \begin{bmatrix} R_{3\times3} & r_{3\times1} \\ 0_{1\times3} & 1_{1\times1} \end{bmatrix}$$

donde *r* es la traslación de la transformación y es por ello el uso de la instrucción

```
origen.translation = A(1:3,4)';
```

Y para la rotación es necesario utilizar la submatriz *R* para encontrar el ángulo de rotación θ definido por

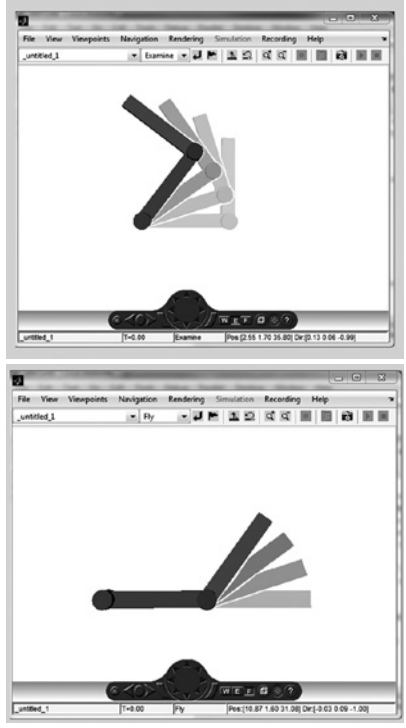
$$\theta = \cos^{-1}\left(\frac{Tr(R)-1}{2}\right) = \cos^{-1}\left(\frac{r_{11}+r_{22}+r_{33}-1}{2}\right)$$

y el vector *k* es dado por

$$k = \frac{1}{2\sin\theta}\begin{bmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12} \end{bmatrix}$$

Ambos se encuentran codificados al final del Listado 8. En las Figura 3 se muestran gráficamente los resultados obtenidos con el robot plano.

Figura 3. Movimientos robot plano
Figure 3. Planar robot movements

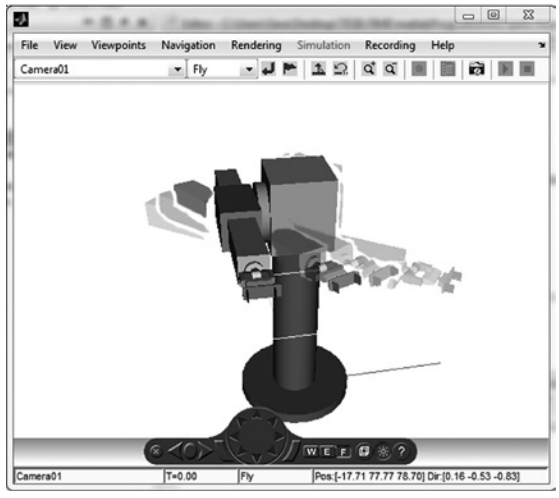


Posterior al emparejamiento de los sistemas de coordenadas se pueden agregar utilerías para guardar los parámetros y definir de esta manera la matriz de transformación homogénea. En la aplicación de robots sólidos seriales, como el de configuración plana presentada en esta sección se encontró un comportamiento muy interesante que discutiremos a continuación. Una vez definido los sistemas de coordenadas entre eslabones consecutivos, la cinemática directa se obtiene alternando las matrices de transformación homogénea encontradas con matrices similares con parámetros cero excepto aquél cuyo eslabón define su naturaleza variable como lo es el parámetro d si es prismático o lineal, o en su caso θ si lo define uno de rotación. Es decir, si definimos los parámetros que definen la transformación T_i entre un sistema de coordenadas actual, *e.g.*, i –ésimo, con el anterior, *i.e.*, $i-1$, entonces, para representar la variación del eslabón del sistema actual es necesario postmultiplicar por la matriz homogénea cuyos parámetros son cero excepto el que define la variable de unión. Por ejemplo, si es una variable de unión prismática la transformación se define por $L_i = A(0,0,a,0)$, y si es de rotación la definición es dada por $L_i = A(\theta,0,0,0)$. Esto último se realiza debido a que los sistemas de coordenadas encontrados son objetos matemáticos independientes del manipulador en sí y permanecen fijos en el espacio, por lo que las matrices L_i le permiten moverse simultáneamente, *i.e.*, fijos, con el eslabón respectivo. Este es un concepto que en lo particular los autores no han visto ninguna referencia que lo mencione por lo que queda considerada como un resultado aunque sencillo, innovador. De esta forma la cinemática directa para un manipulador de n grados de libertad es dada por la transformación T_0^n, que relaciona al último sistema de coordenadas con la base, y que es definida como

$$T_0^n(q_1,\dots,q_n) = T_1L_1(q_1)T_2L_2(q_2)\cdots T_nL_n(q_n)$$

Lo que representa una cadena de la cinemática directa de $2n$ sistemas de coordenadas con dependencia de las n variables generalizadas $q_i, i = 1,\dots,n$ por parte de las matrices L_i mientras que las respectivas T_i obtenidas con nuestro método permanecen constantes. Finalmente, en la Figura 9 se muestra un manipulador Stanford de 6 grados de libertad estudiado en [12] al cual se le aplicó la metodología presentada en este artículo y del que se obtuvieron resultados muy satisfactorios.

Figura 4. Robot Stanford
Figure 4. Stanford Robot



3. CONCLUSIONES

La metodología presentada ha permitido obtener resultados muy favorables en el empleo de mecanismos, en especial robots 3D, con dimensiones o medidas desconocidas, o bien, con geometrías muy complejas que dificulta la aplicación de conceptos de la cinemática tanto directa como inversa por la dificultad natural de determinar los pivotes de giro en cada una de sus uniones y la ubicación de los sistemas de coordenadas en sus eslabones, aunando a esto las dificultades propias en la interpretación de las diferentes perspectivas de los gráficos que se puedan obtener para analizar cada uno de los casos.

REFERENCIAS

[1] J.A. Esquivel Cárdenas *Apuntes de robótica*. Facultad de Ingeniería Mecánica y Eléctrica. Universidad Autónoma de Coahuila, Monclova, México, 2012.

[2] J. Craig *Introduction to Robotics: Mechanics and Control*, 2nd E. Addison Wesley, Longman 1989.

[3] W.M. Spong and M.Vidyasagar, *Robot Dynamics and Control*, Wiley, 1989.

[4] F.L. Lewis, M.D. Darren and C.T. Abdallah. *Manipulator Control Theory and Practice*. Marcel Dekker. 2006.

[5] T.R. Kurkess, *Robotics and Automation Handbook*. CRC Press. 2005.

[6] L.W. Tsai, *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*, Ed. Wiley and Sons, 1999.

[7] U. Zaldívar Colado, *Robótica Asistida por Teleoperación y Realidad Virtual*. Sección de Computación Departamento de Ingeniería Eléctrica del CINVESTAV, 2003.

[8] D. Araujo Díaz, D. *El Placer de Programar con VRML 2.0*, México, D.F., 2006

[9] D. Araujo Díaz, D. *Sistema de Visión Artificial para el Laboratorio de Robótica Virtual*, Sección de Computación-Departamento de Ingeniería Eléctrica del CINVESTAV, México, D.F., 2002.

[10] E. De Alarcón Álvarez y N. Parés I. Burgués. *Manual Práctico de VRML 2.0*; Ed. Biblioteca Técnica de Programación, España, 2000.

[11] D.H. Stolfi Rosso y S. Gálvez Rojas, *Mundos Virtuales 3D con VRML97*. Dpto. de Lenguajes y Ciencias de la Computación - E.T.S. de Ingeniería Informática - Universidad de Málaga.

[12] G. Rincón Maltos, "Programación en VRML y Matlab para la Generación de matrices de Transformación Homogénea en el Estudio de la Robótica Virtual," Tesis de Maestría. Facultad de Ingeniería Mecánica y Eléctrica, Universidad Autónoma de Coahuila. Monclova, México. Dic. 2012.

[13] G. Visansay Fernandez, *Laboratorio Virtual de Física en Matlab/Simulink con Tecnología VRML*. Facultad de Informática de la Universidad de Cienfuegos "Carlos Rafael Rodríguez" Cienfuegos, Cuba. 2008.

[14] D.O. Barragán Guerrero, Manual de Interfaz Gráfica de Usuario en MATLAB: parte 1. Ecuador 2008;

[15] MathWorks Inc. MATLAB7 *Creating Graphical User Interfaces*, 2009

[16] MathWorks Inc. Help Browser; 2010.

[17] M Vargas Villanueva, *Tutorial de Introducción a MATLAB*. *<http://grupo.unavirtual.una.ac.cr /mahara/artefact/file/download.php?file=6824&view=1085>* [Acceso 20 de Noviembre 2012]

[18] J.M. Hobaica Alvarado, *J.M. Señales y Sistemas en MATLAB y LabVIEW*. Rice University, Houston, Texas. 2012. *<http://cnx.org/content/col11361/1.4/>* [Acceso 20 de Noviembre 2012]

[19] E. Corrado Padilla, J.J. Delgado y S. Castañeda. *Tecnologías de Realidad Virtual: Modelo Edificio Inteligente*. Depto. de Cómputo -CICESE, Ensenada, B.C. Disponible en: *<http://telematica.cicese. mx/computo/super/cicese2000/realvirtual/>* [Acceso 20 de Noviembre 2012]

[20] A.M. Khamis Rashwan, *Interacción Remota con Robots Móviles Basada en Internet*. Departamento de Ingeniería de Sistemas y Automática de la Universidad Carlos III de Madrid, Madrid, 2003.

[21] L. López Pacheco, V.M. Olguín. *Scorbot ER-VII virtual: modelado matemático y control de movimien-to*. Centro de Investigación en Tecnologías de Información y Sistemas de la Universidad Autónoma del Estado de Hidalgo. Pachuca de Soto, México. 2006.

[22] G. Velez Jhan, *Curso VRML. Arquitectura y Planeamiento* Urbano - Facultad de Ingeniería de la Universidad Nacional de la Patagonia "San Juan Bosco." *<http://www.macoco.8m.com/indice.htm>* [Acceso 20 de Noviembre 2012]

[23] B. Shneiderman, *Designing the Users Interface, Strategies for effective Human-Computer Interaction*. 3a edición, Ed. Addison-Wesley, U.S.A., 1998.