Parametrizaciones de Movimientos Rígidos en 3D para Visión Artificial

María Lorena Bergamini, Jorge Alejandro Kamlofsky
CAETI – Facultad de Tecnología Informática
Universidad Abierta Interamericana
{maria.bergamini, jorge.kamlofsky}@uai.edu.ar

Resumen

La visión estereoscópica se basa en captar dos imágenes de una misma escena mediante un par de sensores. Esta habilidad permite a muchos seres vivos realizar representaciones tridimensionales de su entorno. Una de las técnicas usuales de visión robótica implementa esta habilidad analizando pares de imágenes obtenidas de un mismo objeto, y determinando un conjunto de puntos homólogos. La posición relativa de las cámaras se modela con una transformación rígida en el espacio: rotación y traslación. Por otra parte, estas transformaciones también son fundamentales al implementar sistemas automáticos de localización, orientación y seguimiento de objetos moviéndose en el espacio. Frente a las diversas alternativas de modelado y parametrización de las transformaciones rígidas, se pretende analizarlas desde un enfoque unificado, proponer variantes mejoradoras y medir y comparar su desempeño computacional.

1. Introducción

La estimación automática de la posición de un objeto rígido moviéndose en el espacio tridimendional, a partir de imágenes bidimensionales del objeto, o a partir de datos proporcionados por sensores ubicados en el mismo, constituye un problema de fundamental importancia en la tecnología actual. Y junto con la tarea de determinar la posición, surge la necesidad de gobernar y controlar el movimiento automáticamente (Murray et al., 1994).

El desarrollo de herramientas matemáticas para cumplir esta tarea ha sido foco de investigaciones de matemáticos e informáticos dedicados a visión computacional, videojuegos y robótica. Estas aplicaciones requieren formas eficientes de representar transformaciones de coordenadas en el espacio; sobre todo en operación y control online de cámaras o dispositivos robóticos.

Un movimiento rígido o desplazamiento en el espacio está compuesto por una traslación y una rotación. El conjunto de movimientos de rotaciones constituye el *grupo especial ortogonal* de tres dimensiones, SO(3). Este grupo algebraico es isomorfo a las matrices de orden 3, ortogonales y con determinante positivo.

Cualquier configuración de un objeto rígido que es libre de rotar en relación a un sistema de referencia fijo puede ser identificada con una matriz ortogonal con determinante positivo, llamada matriz de rotación. Por eso, el grupo SO(3) se lo conoce como espacio de configuración (Murray et al., 1994).

Existen diversas parametrizaciones para determinar orientaciones o configuraciones de un objeto (Diebel, 2006, Kuipers, 2002). Las coordenadas más naturales son el eje de rotación y el ángulo.

Una alternativa es el uso de los llamados ángulos de Euler para determinar la configuración rotacional de objetos, principalmente en posicionamiento relativo de vehículos aeroespaciales. Son tres ángulos que definen rotaciones con respecto a los tres ejes coordenados. Existen múltiples convenciones para el orden en que se consideran los ejes, y por lo tanto, múltiples conjuntos de ángulos que representan la misma configuración.

Otras herramientas matemáticas para el mismo fin son las matrices de rotación, y los cuaterniones.

Las relaciones entre ángulos de Euler, matrices de rotación y cuaterniones son clásicas, y existe una amplia bibliografía referente al tema. En particular, Serrano et al. (2014) realizan un profundo estudio de la justificación geométrica y algebraica de cada una de las representaciones.

Varios problemas en visión computacional y robótica involucran la tarea de encontrar los parámetros de la configuración óptima, lo que implica un problema de optimización en SO(3). Por ejemplo, tal problema surge en la determinación de la posición de una cámara, conociendo un conjunto de puntos en imágenes tomadas por ésta.

Dados puntos homólogos, puestos en correspondencia por medio de algoritmos de detección de características en un par de imágenes, se plantea el problema de estimar la posición relativa de los puntos de toma. Dicha posición puede establecerse como un elemento en SO(3), y se

calcula de forma tal de minimizar el error de predicción en los puntos homólogos. Sarkis y Diepold estudian este problema, y proponen un algoritmo numérico para optimizar en SO(3) (Sarkis Diepold, 2012).

Es fundamental, a la hora de diseñar algoritmos de localización, seguimiento y/o control de objetos, utilizar la mejor formulación, desde el punto de vista de minimizar el costo computacional, y la propagación de errores de punto flotante.

Existe una extensa literatura sobre las distintas parametrizaciones, pero muy pocas que muestren un tratamiento unificado (Diebel, 2006; Serrano et al., 2014). Más aún, casi no hay trabajos que hagan una comparación computacional desde el punto de vista práctico.

El objetivo del presente trabajo es en primer lugar, exponer unificadamente las distintas alternativas de representación de transformaciones rígidas en el espacio, la mayoría de ellas son las usuales, y para algunas se sugieren modificaciones. Además, se pretende evaluar el desempeño computacional de las diferentes parametrizaciones del grupo especial ortogonal, usando funciones de cálculo numérico y algoritmos de optimización genéricos en librerías de software de computación numérica.

En la siguiente sección se presentan las distintas formulaciones que se analizarán; luego se expone el problema de la determinación de parámetros óptimos de la transformación que relaciona dos conjuntos de puntos. Seguidamente se muestran los resultados experimentales; y el trabajo finaliza con la discusión de resultados y conclusiones.

2. Formulación de rotaciones en el espacio

La posición de un objeto rígido en el mundo tridimensional, con respecto a un sistema de referencia fijo, puede descomponerse en una traslación y una rotación. Matemáticamente, una traslación es

$$\boldsymbol{r} = \boldsymbol{p} + \boldsymbol{t} \tag{1}$$

donde $\boldsymbol{p}$ es la posición actual del objeto (del punto de referencia del mismo), $\boldsymbol{r}$ es la posición luego de la traslación y $\boldsymbol{t}$ es el vector de desplazamiento.

Más desafiante es la formulación matemática de una rotación. En cualquier dimensión, la rotación de un punto $\boldsymbol{p}$ se modeliza como

$$\boldsymbol{r} = M\,\boldsymbol{p} \tag{2}$$

donde M es una matriz ortogonal, es decir, que verifica $M^{-1} = M^t$, y con determinante positivo (las matrices ortogonales con determinante negativo representan rotaciones seguidas de una inversión, que no corresponde a un movimiento de un cuerpo rígido). Esta matriz M tiene un autovalor 1 y el autovector asociado es el eje de la rotación.

En R^3, una rotación de un ángulo θ alrededor de un eje $\boldsymbol{w} = (x,y,z)^t$ (de módulo 1) se puede modelar con la matriz

$$M = \begin{bmatrix} \mathrm{C} + (1-\mathrm{C})x^2 & xy(1-\mathrm{C}) - z\mathrm{S} & xz(1-\mathrm{C}) + y\mathrm{S} \\ xy(1-\mathrm{C}) + z\mathrm{S} & \mathrm{C} + (1-\mathrm{C})y^2 & yz(1-\mathrm{C}) - x\mathrm{S} \\ xz(1-\mathrm{C}) - y\mathrm{S} & yz(1-\mathrm{C}) + x\mathrm{S} & \mathrm{C} + (1-\mathrm{C})z^2 \end{bmatrix} \tag{3}$$

donde $C = cos(\theta)$, $S = sen(\theta)$.

En esta representación ocurren singularidades, cuando $\theta = 0$; no está bien definido el eje de rotación en tal caso.

Nótese que la condición de módulo 1 para el vector $\boldsymbol{w}$ puede evitarse si se escribe el eje en coordenadas esféricas ϕ y η,

$$\boldsymbol{w} = (\cos(\phi)\mathrm{sen}(\eta), \mathrm{sen}(\phi)\mathrm{sen}(\eta), \cos(\eta))^t \tag{4}$$

Así, la matriz M depende de los tres parámetros, θ, ϕ y η. Esta formulación será denotada 3'. Esta representación también tiene singularidades, por ejemplo, el vector $\boldsymbol{w} = (0,0,1)^t$ no tiene parametrización única.

Por otro lado, es sabido que una rotación puede descomponerse en tres rotaciones básicas, alrededor de los ejes coordenados. Así,

$$M = R_x(\alpha)\, R_y(\beta)\, R_z(\gamma) \tag{5}$$

representa la composición de una rotación de un ángulo γ alrededor del eje z, seguida de una rotación de un ángulo β alrededor del eje y, y finalmente una rotación con un ángulo α alrededor del eje x, donde

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \mathrm{sen}(\alpha) \\ 0 & -\mathrm{sen}(\alpha) & \cos(\alpha) \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & -\mathrm{sen}(\beta) \\ 0 & 1 & 0 \\ \mathrm{sen}(\beta) & 0 & \cos(\beta) \end{bmatrix} \tag{6}$$

$$R_z(\gamma) = \begin{bmatrix} \cos(\gamma) & \mathrm{sen}(\gamma) & 0 \\ -\mathrm{sen}(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

son las matrices básicas de rotación alrededor de los ejes de la referencia. Los ángulos (α,β,γ) son los *ángulos de Euler* de la secuencia ZYX. Como se mencionó anteriormente, existen otras convenciones para el orden en que se componen los giros alrededor de los ejes, dando otras ternas de ángulos para la misma rotación.

La matriz dada en (5) es una matriz ortogonal, y se corresponde con una rotación de un ángulo $\theta = 2cos^{-1}\left(\sqrt{1+m_{11}+m_{22}+m_{33}}/2\right)$ alrededor de un eje $\boldsymbol{w}$ en la dirección $(m_{32}-m_{23}, m_{13}-m_{31}, m_{21}-m_{12})^t$, donde m_{ij} es el elemento ij de M.

Dada una matriz de rotación M, la descomposición en ángulos de Euler no es única, por lo que surgen singularidades al tratar de resolver ese problema. No existe una relación biunívoca entre una transformación de rotación y una terna de ángulos (α,β,γ); por lo que esta parametrización no es isomorfa a SO(3).

Escribiendo el eje de rotación en coordenadas esféricas, se puede escribir $\boldsymbol{w} = R_z(\phi)R_y(\eta)\boldsymbol{z}$, siendo $\boldsymbol{z} = (0,0,1)^t$. Luego, es fácil deducir que la rotación de un ángulo θ alrededor de $\boldsymbol{w}$ se representa con la matriz

$$M = R_z(\phi)R_y(\eta)R_z(\theta)R^t_y(\eta)R^t_z(\phi) \qquad (7)$$

Esta matriz, también ortogonal, depende de los tres ángulos θ, ϕ y η, y por consiguiente, es la misma matriz que la obtenida en (3').

Existe otra forma de parametrizar de SO(3) que no utiliza matrices de rotación, y no presenta singularidades. Se basa en los cuaterniones y las operaciones definidas en este conjunto de números hipercomplejos (Sahu et al., 2008; Pham et al., 2010; Chao et al., 2006).

Un cuaternión puede pensarse como un elemento de R^4. Se lo denota $\boldsymbol{q} = (a, x, y, z)$, o también $\boldsymbol{q} = (a,\boldsymbol{v})$, donde $a \in R$ se denomina parte real y $\boldsymbol{v} \in R^3$ parte vectorial del cuaternión.

El producto de dos cuaterniones $\boldsymbol{q_1} = (a_1, x_1, y_1, z_1)$ y $\boldsymbol{q_2} = (a_2, x_2, y_2, z_2)$, es el cuaternión $\boldsymbol{q_3} = \boldsymbol{q_1}.\boldsymbol{q_2} = (a_3, x_3, y_3, z_3)$ con

$$\begin{aligned} a_3 &= a_1.a_2 - x_1.x_2 - y_1.y_2 - z_1.z_2 \\ x_3 &= a_1.x_2 + x_1.a_2 + y_1.z_2 - z_1.y_2 \\ y_3 &= a_1.y_2 - x_1.z_2 + y_1.a_2 + z_1.x_2 \\ z_3 &= -a_1.z_2 + x_1.y_2 - y_1.x_2 + z_1.a_2 \end{aligned} \qquad (8)$$

Si se representan los cuaterniones con $\boldsymbol{q_1} = (a_1, \boldsymbol{v_1})$ y $\boldsymbol{q_2} = (a_2, \boldsymbol{v_2})$, el producto también se puede escribir

$$\boldsymbol{q_3} = (a_1a_2 - \boldsymbol{v_1}\cdot\boldsymbol{v_2}, a_1\boldsymbol{v_2} + a_2\boldsymbol{v_1} + \boldsymbol{v_1}\times\boldsymbol{v_2}) \qquad (9)$$

El conjugado del cuaternión $\boldsymbol{q} = (a, \boldsymbol{v})$ es $\boldsymbol{q}^* = (a, -\boldsymbol{v})$. La norma se define como $|\boldsymbol{q}| = (\boldsymbol{q}.\boldsymbol{q}^*)^{1/2} = (a^2 + x^2 + y^2 + z^2)^{1/2}$, y un cuaternión unitario es aquel que tiene norma 1.

Todo cuaternión unitario se puede escribir en la forma $\boldsymbol{q} = (\cos(\theta/2), \boldsymbol{w}.\text{sen}(\theta/2))$ con $\boldsymbol{w}$ vector unitario en R^3.

Las posiciones de objetos en el espacio pueden representarse como cuaterniones con parte real nula, llamados cuaterniones puros.

Los cuaterniones unitarios constituyen una formulación matemática para representar las orientaciones y rotaciones de objetos en tres dimensiones. En Torres del Castillo (1999), puede encontrarse una justificación geométrica de la relación entre cuaterniones y rotaciones.

Siendo $\boldsymbol{p}$ un cuaternión puro, y $\boldsymbol{q}$ un cuaternión unitario, $\boldsymbol{q} = (\cos(\theta/2), \boldsymbol{w}.\text{sen}(\theta/2))$, la transformación

$$\boldsymbol{r} = \boldsymbol{q}.\boldsymbol{p}.\boldsymbol{q}^* \qquad (10)$$

da el punto $\boldsymbol{r}$ que es la imagen de $\boldsymbol{p}$ por la rotación con eje $\boldsymbol{w}$ un ángulo θ.

Ese producto doble puede calcularse mediante la aplicación sucesiva de las expresiones (8), a partir de las cuatro componentes de $\boldsymbol{q}$, a la que llamaremos formulación 10.8; o la expresión (9), a partir de la parte real y la parte vectorial de $\boldsymbol{q}$, que llamaremos formulación 10.9.

Escribiendo el cuaternión $\boldsymbol{q} = (a, x, y, z)$, y usando el hecho que es unitario, $a^2+x^2+y^2+z^2=1$, la ecuación (10) se puede reescribir como $\boldsymbol{r} = M\,\boldsymbol{p}$, siendo M la matriz

$$\begin{bmatrix} 2a^2+2x^2-1 & 2(xy-az) & 2(ay+xz) \\ 2(xy+az) & 2a^2+2y^2-1 & 2(yz-ax) \\ 2(xz-ay) & 2(ax+yz) & 2a^2+2z^2-1 \end{bmatrix} \qquad (11)$$

donde, $\boldsymbol{p}$ ahora se toma como un punto en R^3.

Nótese que las formulaciones 10.8, 109 y 11 son polinómicas, es decir, involucran sólo operaciones de suma y producto.

Escribiendo el cuaternión $\boldsymbol{q} = (\cos(\theta/2), \boldsymbol{w}.\text{sen}(\theta/2))$, siendo $\boldsymbol{w}$ unitario, la ecuación (10) con (9) resulta

$$\boldsymbol{r} = \cos(\theta)\boldsymbol{p} + \text{sen}(\theta)\,\boldsymbol{w}\times\boldsymbol{p} + (1-\cos(\theta))(\boldsymbol{w}\cdot\boldsymbol{p})\,\boldsymbol{w} \qquad (12)$$

que es la conocida fórmula de Rodrigues.

En este caso, si el vector unitario $\boldsymbol{w}$ se escribe en coordenadas esféricas, se obtiene una nueva formulación, que será denotada 12'.

Existe también una parametrización que resulta ser una variante de las formulaciones 3 y 12. Se obtiene parametrizando cada rotación con el llamado vector de rotación $\boldsymbol{u} = (X,Y,Z)^t$, de forma tal que $\theta = |\boldsymbol{u}|$, y el eje de rotación es paralelo a $\boldsymbol{u}$. Así, $\boldsymbol{u} = \theta\,\boldsymbol{w}$. Tomando estas variables, $(X,Y,Z)^t$, como variables libres, la rotación se puede calcular a partir de la ecuación 3 o la ecuación 12. En este último caso, se obtiene

$$\begin{aligned} \boldsymbol{r} = cos|\boldsymbol{u}|\,\boldsymbol{p} + sen|\boldsymbol{u}|\,\left(\frac{\boldsymbol{u}}{|\boldsymbol{u}|}\times\boldsymbol{p}\right) + \\ (1-cos|\boldsymbol{u}|)(\frac{\boldsymbol{u}}{|\boldsymbol{u}|}\cdot\boldsymbol{p})\frac{\boldsymbol{u}}{|u|} \end{aligned} \qquad (13)$$

Esta parametrización tampoco es isomorfa a SO(3), ya que distintos parámetros $\boldsymbol{u}$ pueden representar la misma rotación.

3. Determinación de los parámetros de una transformación

Se plantea el problema de determinar los parámetros de una transformación rígida que relaciona dos conjuntos de puntos. Específicamente, sean los puntos $\boldsymbol{p_i}$ y $\boldsymbol{r_i}$, con $i = 1,\ldots,N$, de los cuales se espera que verifiquen

$$\boldsymbol{r_i} = f(\boldsymbol{p_i}, \Omega) \tag{14}$$

para cada i, donde $f(\cdot, \Omega)$ es una transformación rígida con parámetros Ω. Este problema surge, por ejemplo, cuando se tienen dos imágenes de la misma escena, tomadas desde dos posiciones distintas. Los pares $(\boldsymbol{p_i},\boldsymbol{r_i})$ son puntos homólogos, y la transformación f determina la posición relativa de los puntos de toma.

Siendo que el cálculo de puntos homólogos conlleva errores de precisión e incertidumbre, el problema de hallar los parámetros Ω de f se plantea como un problema de optimización

$$\min_{\Omega \in \Lambda} \; error(\boldsymbol{r_i} - f(\boldsymbol{p_i}, \Omega)) \tag{15}$$

Las distintas formulaciones para la transformación f, implicará distinto costo computacional, y distinta precisión en el cálculo de los parámetros. En los tests expuestos en la sección de resultados experimentales, consideramos que f es sólo una rotación.

La tabla 1 enumera los parámetros necesarios para cada una de las formulaciones descritas en la sección anterior; así como el espacio de factibilidad Λ.

Tabla 1: Parámetros usados en cada formulación

Ec.	Parámetros	Λ
3	(θ,x,y,z)	$\{(\theta,x,y,z): \lvert(x,y,z)\rvert=1\}$
3'	(θ,η,ϕ)	R^3
5	(α,β,γ)	R^3
7	(θ,η,ϕ)	R^3
10.8	$\boldsymbol{q}=(a,x,y,z)$	$\{\boldsymbol{q}=(a,x,y,z): \lvert\boldsymbol{q}\rvert=1\}$
10.9	$\boldsymbol{q}=(a,\boldsymbol{v})$	$\{\boldsymbol{q}=(a,\boldsymbol{v}): a\in R, \boldsymbol{v}\in R^3, \lvert\boldsymbol{q}\rvert=1\}$
11	$\boldsymbol{q}=(a,x,y,z)$	$\{\boldsymbol{q}=(a,x,y,z): \lvert\boldsymbol{q}\rvert=1\}$
12	$(\theta,\boldsymbol{w})$	$\{(\theta,\boldsymbol{w}): \theta\in R, \boldsymbol{w}\in R^3, \lvert\boldsymbol{w}\rvert=1\}$
12'	(θ,η,ϕ)	R^3
13	(X,Y,Z)	R^3

Todas las formulaciones tienen 3 o 4 parámetros. Cuando la cantidad de parámetros es 4, el espacio de factibilidad está descrito por alguna restricción (normalización de vectores) que reduce en uno los grados de libertad. Entonces, el grado de libertad es siempre 3.

4. Resultados experimentales

Se han realizado distintos tests para comparar experimentalmente las ventajas y desventajas de las distintas formulaciones de transformaciones rígidas.

Fueron implementados en el software de cálculo numérico Scilab v.5.5.2 (Scilab, 2015), en una PC con procesador Intel(R) Core (TM) i5 CPU de 3.20GHz con 3.17Gb de memoria RAM.

El primer experimento pretende medir el uso de CPU necesario para efectuar una determinada cantidad de rotaciones sobre una nube de puntos. Para ello, se han tomado aleatoriamente un conjunto de rotaciones, dadas por los parámetros ángulo-eje. Además, se han tomado aleatoriamente una nube de puntos en el espacio.

Se realizaron 5000 rotaciones sobre una cantidad variable de puntos. La figura 1 muestra el uso de CPU para las distintas formulaciones. En la leyenda de la figura se identifica el número de ecuación correspondiente a cada serie de datos. No se incluyen resultados de la formulación (13), ya que con ella se realizan los mismos cálculos que la formulación (12).

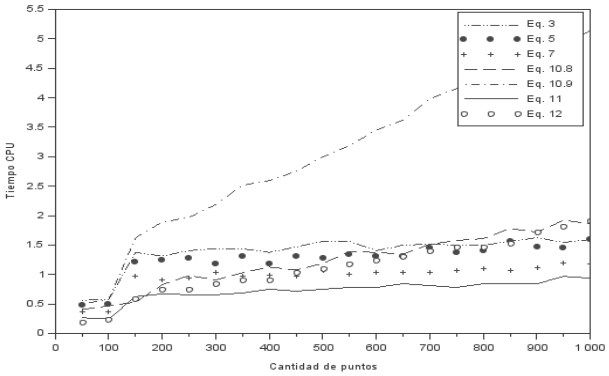

Figura 1: Tiempos de CPU para computar rotaciones

De esta figura se puede concluir que la alternativa notablemente más costosa computacionalmente es calcular las rotaciones usando cuaterniones, de acuerdo a la fórmula (10), con el producto de cuaterniones definidos en (9). También merece ser remarcado que la formulación que ejecuta la tarea en menor tiempo es la dada en la ecuación (11), que supone el uso de cuaterniones, con el producto en forma matricial. Todas las demás formulaciones son prácticamente equiparables en cuanto al tiempo de computación, ya que muestran la misma tendencia.

Otro análisis que se realizó fue enfocado a determinar la pérdida de precisión numérica sufrida al usar las distintas formulaciones.

Para ello, se realizaron S rotaciones sucesivas sobre un conjunto de N puntos. Se aplica iterativamente la transformación $\boldsymbol{p} = f(\boldsymbol{p})$, a partir de un conjunto inicial $\boldsymbol{p_0}$ de N puntos, donde f se representa por las formulaciones

mencionadas anteriormente. Las S rotaciones son tales que, con precisión infinita, $\boldsymbol{p_0} = f^S(\boldsymbol{p_0})$. El error acumulado se calcula como

$$|\boldsymbol{p_0} - f^S(\boldsymbol{p_0})|_\infty = \max_i \left\{ \left(\boldsymbol{p_0} - f^S(\boldsymbol{p_0}) \right)_i \right\} \qquad (16)$$

Scilab maneja números reales con el sistema de punto flotante establecido en el estándar IEEE 754, con precisión relativa 2^{-52}. La tabla 2 resume los resultados obtenidos. En la misma, se muestra el error promedio relativo alcanzado con cada formulación (el promedio se toma sobre un conjunto de tests con distintos S y N).

Tabla 2: Errores relativos de precisión acumulados

Ec.	3	5	7	10.8	10.9	11	12
Error	1.0	2.4	1.8	0.6	0.5	1.3	1.4

Los resultados indican que no hay marcada diferencia en la pérdida de precisión entre las distintas formulaciones usadas, a pesar de que algunas involucran sólo operaciones polinómicas (productos y sumas), y otras requieren gran cantidad de evaluación de funciones trigonométricas. Todas sufren una pérdida de precisión del mismo orden.

Sin embargo, puede notarse que las ecuaciones (10.8) y (10.9), que son básicamente polinómicas, muestran el menor error. Y, por otra parte, la formulación con los ángulos de Euler (ecuación 5), involucrando senos y cosenos de tres ángulos distintos, presenta el error más grande.

Por otro lado, se realizó un test para resolver el problema de encontrar los parámetros de la transformación que relaciona dos conjuntos de puntos homólogos. Se tomaron, en principio, 20 pares de puntos homólogos ($\boldsymbol{p_i}$, $\boldsymbol{r_i}$), relacionados por una transformación prefijada (Ω_c conocido). A los puntos $\boldsymbol{r_i}$ se les agregó una perturbación gaussiana.

Se realizó la optimización de la ecuación (15) usando la función *optim* de Scilab, que implementa el algoritmo Quasi-Newton con BFGS. Las restricciones de módulo 1 en las formulaciones 3, 10.8, 10.9, 11 y 12 se modelaron con una penalidad en la función objetivo.

Al tratarse de un problema de optimización no lineal, el punto inicial tiene gran influencia en el resultado y en la cantidad de iteraciones necesarias para alcanzar convergencia. Para cada prueba, se toma un ángulo inicial θ_0 y un eje inicial $\boldsymbol{w_0}$, aleatoriamente en un intervalo adecuado para cada parámetro; y luego se traducen esos valores a los parámetros iniciales correspondientes en cada formulación.

En todos los casos, (excepto los casos singulares de los cuales se habla más adelante) el criterio de terminación fue alcanzado satisfactoriamente (con los parámetros de convergencia por defecto de la función *optim* de Scilab, excepto el número de evaluaciones de función objetivo -*nap*-, que fue elevado a 2000).

En la tabla 3 se reportan los tiempos promedios empleados para cada formulación, así como la desviación estándar. El promedio se calcula sobre los test realizados con distintos puntos iniciales.

Tabla 3: Tiempos de CPU

Ec.	**Tiempo CPU**	**Desv**	**Iters**
3	0.252	0.089	43
3’	0.064	0.016	13
5	0.063	0.019	14
7	0.068	0.020	14
10.8	0.083	0.014	21
10.9	0.083	0.017	21
11	0.082	0.020	22
12	0.214	0.057	41
12’	0.067	0.021	13
13	0.070	0.024	14

Puede notarse que las formulaciones que implican mayor costo computacional son la que se basa en la matriz dada en (3), parametrizada con ángulo-eje, y la basada en la fórmula de Rodrigues con los mismos parámetros. Estas parametrizaciones implican un espacio de factibilidad definido por una restricción de normalización, que se introduce como una penalidad en la función objetivo.

También debe remarcarse que en dichas formulaciones, si se utilizan las coordenadas esféricas para el eje, evitando así la restricción de normalización, el tiempo computacional decrece notablemente, resultando de esta manera las formulaciones óptimas, desde el punto de vista del tiempo de CPU usado.

La formulación que se basa en ángulos de Euler implica un tiempo equivalente.

Asimismo, las formulaciones basadas en producto de cuaterniones (10.8, 10.9 y 11) igualmente tienen una restricción de normalización, y el tiempo implicado es levemente mayor a las formulaciones de mejor desempeño, pero no tan elevadas como las (3) y (12). Esto es porque la parametrización con cuaterniones sí es un isomorfismo en SO(3), y por lo tanto no presenta redundancia.

Es interesante también hacer notar la relación entre tiempo de CPU y cantidad de iteraciones del algoritmo de optimización, así como cantidad de evaluaciones de la función objetivo por iteración; como se ven en la tabla 4 (en valores promedios).

Tabla 4: Tiempo CPU y evaluación de objetivo por iteración

Ec.	Tiempo CPU/iter 10^3	Evals/iter
3	5.84	4.0
3’	4.75	3.6
5	4.70	3.7
7	4.90	3.6
10.8	3.99	2.4
10.9	4.03	2.4
11	3.80	2.6
12	5.22	3.5
12’	4.80	3.5
13	5.21	4.0

Se observa que las alternativas basadas en cuaterniones son las que implica menores tiempos y evaluaciones por iteración. Nuevamente, esto puede adjudicarse a que son problemas polinómicos, que no suponen el uso de funciones trigonométricas.

Excepto las formulaciones basadas en cuaterniones, todas presentan casos singulares. Por ejemplo, cuando el eje de rotación es $\boldsymbol{w} = (0,0,1)^t$, no están bien determinados los parámetros en las ecuaciones 3’, 7 y 12’. En los resultados, cuando el eje de rotación es tal $\boldsymbol{w}$, o uno cercano, el número de iteraciones necesarias en la función *optim* se eleva notablemente..

5. Discusión

De los test realizados se puede concluir, por un lado, que en cuanto al tiempo de CPU necesario para realizar un conjunto de transformaciones, es preferible utilizar la parametrización con cuaterniones, y expresar la transformación matricialmente. Esto implica sólo operaciones de suma y producto, frente a otras alternativas que suponen uso de funciones trigonométricas. Utilizar cuaterniones con el doble producto definido a partir de la parte real y parte vectorial del cuaternión es la peor opción. Esto tiene su explicación en que se ejecuta dos veces la función que multiplica cuaterniones, con lo cual, los datos se operan dos veces.

El test realizado para medir la pérdida de precisión numérica muestra que, si bien todas las formulaciones arrojan una pérdida de precisión del mismo orden, el uso de cuaterniones resulta la parametrización más confiable.

En el problema de hallar los parámetros óptimos que relacionan dos conjuntos de puntos, se concluye que es preferible parametrizar el espacio de transformaciones con tres parámetros, ya que la adición de la penalidad en la función objetivo eleva el tiempo de cómputo. La restricción de normalización siempre puede evitarse parametrizando con coordenadas esféricas el eje. Esta estrategia, según nuestro conocimiento, no es muy frecuentemente usada; quizás porque adiciona una gran cantidad de evaluación de funciones trigonométricas. Sin embargo, nuestros resultados muestran que no representa, en general, un serio inconveniente, desde el punto de vista del tiempo requerido. Quizás sí presentaría dificultades numéricas si los parámetros óptimos son, o están cerca de los valores singulares de la representación. En estos casos siempre se puede cambiar de parametrización o cambiar los ejes de la referencia.

Una de nuestras hipótesis era que los cuatro parámetros de la formulación con ángulo-eje, con restricción de normalización del eje, podría mejorarse con los tres parámetros del vector de rotación (ecuación 13). Sin embargo, nuestros experimentos muestran que no existe ventaja computacional en la práctica.

Referencias

- Chao H., Meng M.Q.-H., Mandal M., Liu P.X. “Robot Rotation Decomposition Using Quaternions”. *Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation*, 2006, pp. 1158 – 1163
- Diebel, J. “Representing attitude: Euler angles, unit quaternions, and rotation vectors”. *Matrix*, *58*, 2006, pp. 15-16.
- Kuipers, J. B. “Quaternions and rotation sequences: A primer with Applications to Orbits, Aerospace and Virtual Reality”, Princeton: Princeton university press, 2002
- Murray R., Li Z., Sastry S. “A Mathematical Introduction to Robotic Manipulation”. CRC Press, Inc. Boca Raton, FL, USA, 1994.
- Pham H.L., Perdereau V., Adorno B., Fraisse P. “Position and Orientation Control of Robot Manipulators Using Dual Quaternion Feedback”. *IROS'10: International Conference on Intelligent Robots and Systems*, 2010, Taipei, Taiwan. IEEE/RSJ, pp.658-663.
- Sahu S., Biswal B., Subudhi B. “A Novel Method for Representing Robot Kinematics using Quaternion Theory”. *IEEE Sponsored Conference on Computational Intelligence, Control And Computer Vision In Robotics & Automation*, 2008, NIT Rourkela http://hdl.handle.net/2080/689
- Sarkis, M., Diepold, K. “Camera-pose estimation via projective Newton optimization on the manifold”. *Image Processing, IEEE Transactions on*, 21(4), 2012, pp. 1729-1741.
- Scilab. *Scilab Enterprises* http://www.scilab.org/, 2015
- Serrano E., Sirne R., La Mura G. “Rotaciones, secuencia aeroespacial y cuaterniones. Una revisión de las relaciones fundamentales*”. Ciencia y Tecnología*, 14, 2014, pp. 11-28.
- Torres del Castillo G.F. “La representación de rotaciones mediante cuaterniones”. *Miscelánea Matemática* 29, 1999, pp. 43-50.