Aplikasi Rotasi Vektor dengan Bilangan Quartenion dan Implementasinya pada Fungsi SLERP

Atika Azzahra Akbar 13514077[1]
Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13514077@std.stei.itb.ac.id

***Abstrak*—Dalam makalah ini akan dibahas sebuah aplikasi dari bilangan quaternion yaitu rotasi vektor. Rotasi vektor menggunakan bilangan quaternion digunakan dalam bidang animasi yaitu pada *computer animation*. Bilangan quaternion merupakan kunci dalam fungsi perpindahan pada computer animation yang dikenal sebagai fungsi SLERP, *spherical linear interpolation,* dan fungsi lainnya yaitu NLERP, *normalized spherical linear interpolation, dan* LERP, *linear interpolation.* Pada makalah ini hanya akan dijelaskan mendetail fungsi SLERP. Kemudian akan dibahas pula pengembangan lanjut dari fungsi SLERP yaitu kurva Bezier.**

***Kata Kunci*—Animasi, Bilangan quaternion, Rotasi vektor, SLERP.**

I. PENDAHULUAN

Dewasa dunia hiburan seperti Hollywood atau Bollywood sedang maju-majunya. Dapat kita saksikan dari begitu banyaknya film yang diproduksi. Tidak terkecuali film animasi. Begitu banyak film animasi yang sedang dan telah diproduksi oleh contohnya Pixar, Disney, Paramount Animation, dan studio-studio lainnya.

Dahulu membuat suatu film animasi sangatlah susah karena dilakukan secara tradisional. Untuk membuat suatu karakter bergerak diperlukan beratus-ratus *frame*. Belum lagi jika film tersebut memiliki warna. Proses pewarnaan dilakukan perlembar *frame.* Setelah tiap *frame* diwarnai versi final dari *frame-frame* tersbut dilanjutkan ke *celluloid* untuk diproduksi ke produk finalnya. Maka dari itu tidak dipungkuri lagi bahwa kemajuan bidang animasi adalah karena ditemukannya *computer animation.*

Dengan adanya *computer animation*, perpindahan gerak pada sebuah karakter tidak perlu lagi digambar pada ratusan frame namun dapat dengan mudah dilakukan dengan interpolasi linear sederhana. Selain itu, rumus fisika biasa tidak dapat menggambarkan perbuahan bentuk yang mata lihat pada objek dinamis, namun dengan adanya *computer animation*, perubahan bentuk sebuah objek dapat diperkirakan secara otomatis. Contohnya seperti bola terpental yang ketika menyentuh tanah bola tersebut berbentuk oval dan ketika mencapai titik tertingginya bola tersebut berbalik membentuk sebuah lingkaran sempurna [1].

Lingkup dari *computer animation* sebenarnya sangat banyak, tidak hanya mengatur pergerakan sebuah gambar namun juga mengatur pencahayaan, sudut kamera, pergerakan kamera, bayangan, suara, warna, dan masih banyak lagi. Pada makalah ini penulis akan membahas tetang salah satu fungsi yang digunakan untuk mengatur pergerakan sebuah karakter pada *computer animation.* Fungsi ini bernama SLERP atau *spherical linear interpolation* adalah implementasi dari bilangan *quartenion.* Lebih tepatnya pada bagian interpolasi bilangan *quartenion.*

II. TEORI DASAR

A. Bilangan Quartenion

Bilangan Quartenion merupakan hasil dari para ilmuwan yang mencoba mencari ekivalen tiga dimensional dari bilangan kompleks. Kemudian, pada tahun 1843 Sir William Rowan Hamilton menemukan apa yang kita sekarang sebut sebagai bilangan quartenion untuk menjawab pertanyaan para ilmuwan [2].

Telah diketahui bahwa bilangan kompleks di bidang R^2 adalah sebagai berikut:

$$z = a + ib$$

Maka, pada awalnya bilangan kompleks di bidang R^3 dituliskan sebagai berikut:

$$z = a + ib + jc$$

Dimana i dan j merupakan bilangan imajiner dengan nilai $i^2 = j^2 = -1$. Namun ketika nilai ini dikalikan dengan sesamanya seperti dibawah ini:

$$z_1 z_2 = (a_1 a_2 - b_1 b_2 - c_1 c_2) + i(a_1 b_2 + b_1 a_2) + j(a_1 c_2 + c_1 a_2) + ijb_1 c_2 + jic_1 b_2$$

Ditemukan objek tidak terdefinisi seperti ij dan ji. Hal ini membuat para ilmuwan bingung, tetapi Sir Hamilton menemukan jawabannya dengan menuliskan persamaan bilangan kompleks di bidang R^3 menjadi persamaan dengan 4-*tuple*.

$$z = a + ib + jc + kd$$

Kemudian untuk perkalian sesamanya menjadi:

$$z_1 z_2 = a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2$$

$+i(a_1b_2 + b_1a_2) + j(a_1c_2 + c_1a_2) + k(a_1d_2 + d_1a_2)$
$+ijb_1c_2 + ikb_1d_2 + jic_1b_2 + jkc_1d_2 + kid_1b_2 +$
kjd_1c_2

Nilai $i^2 = j^2 = k^2 = -1$. Selain itu untuk nilai ij, ik, ji, jk, ki, kj diselesaikan dengan aturan berikut:

$$ij = k \;\; jk = i \;\; ki = j \;\; ji = -k \;\; kj = -i \;\; ik = -j$$

Sehingga persamaan perkalian dari z dapat ditulis menjadi:

$$z_1z_2 = a_1a_2 - (b_1b_2 + c_1c_2 + d_1d_2) + i(a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2) + j(a_1c_2 + c_1a_2 + d_1b_2 - b_1d_2) + k(a_1d_2 + d_1a_2 + b_1c_2 - c_1b_2)$$

Sir Hamilton kemudian mengubah kembali persamaan diatas dengan mensubtitusikan kedua persamaan z dengan penjumlahan nilai skalar dengan sebuah vektor.

$$z_1 = s_1 + v_1 \quad z_2 = s_2 + v_2$$
$$z_1z_2 = s_1s_2 - v_1.v_2 + s_1v_2 + s_2v_1 + v_1 \, x \, v_2$$

Objek z_1 dan z_2 diberikan nama oleh Sir Hamilton 'quartenion' dan nilai imaginer didalamnya 'vektor'.

B. Properti Bilangan Quaternion

Bilangan quartenion yang dicetuskan oleh Sir Hamilton diikuti dengan berbagai properti bilangan tersebut seperti penjumlahan, *product* quaternion, besar nilai quaternion, unit quaternion, *pure* quaternion, konjugasi quaternion, balikan quaternion, aljabar quaternion, dan rotasi vektor menggunakan bilangan quaternion yang akan dibahas lebih dalam disubab berikutnya.

Properti pertama yang akan dibahas pada subab ini adalah penambahan bilangan quaternion. Pertama-tama kita misalkan bahwa q_1 dan q_2 merupakan bilangan quaternion.

$$q_1 = s_1 + ix_1 + jy_1 + kz_1$$
$$q_2 = s_2 + ix_2 + jy_2 + kz_2$$

Setelah itu kedua bilangan dapat dikurang dan ditambahkan seperti aljabar biasa dengan sesama bilangan yang memiliki *terms* yang sama.

$$q_2 \pm q_2 = (s_1 \pm s_2) + i(x_1 \pm x_2) + j(y_1 \pm y_2) + k(z_1 \pm z_2)$$

Properti berikutnya yang akan dibahas adalah product dari bilangan quaternion. Masih menggunakan nilai q_1 dan q_2. Maka nilai product dari keduanya adalah:

$$q_1 = s_1 + v_1 \quad q_2 = s_2 + v_2$$
$$q_1q_2 = s_1s_2 - v_1.v_2 + s_1v_2 + s_2v_1 + v_1 \, x \, v_2$$

Product dari bilangan quaternion tertutup karena kembali menghasilkan bilangan quartenion namun, bilangan komutatif tidaklah komutatif karena bilangan vektornya yaitu v_1 x v_2 memiliki sifat tidak komutatif walaupun bilangan skalar seperti s_2s_1, $v_2.v_1$, dan *product* dari s_2v_1 dan s_1v_2 memiliki sifat komutatif.

Besar suatu bilangan quaternion dapat ditemukan dengan:

$$||q_1|| = \sqrt{{s_1}^2 + {x_1}^2 + {y_1}^2 + {z_1}^2}$$

Properti selanjutnya dari bilangan quaternion adalah unitnya. Unit bilangan quartenion q_1 bersimbol $\widehat{q_1}$, dengan persamaannya sebagai berikut:

$$\widehat{q_1} = \frac{q_1}{||q_1||}$$

Terdapat pula sebuah bilangan quaternion tanpa bilangna skalar. Sir Hamilton menamkannya *pure quaternion* atau bilangan quaternion murni. Maka kita misalkan q_1 dan q_2 merupakan bilangan quaternion murni:

$$q_1 = ix_1 + jy_1 + kz_1$$
$$q_2 = ix_2 + jy_2 + kz_2$$

Ketika q_1 dan q_2 dikalikan, hasilnya bukanlah bilangan quaternion murni lagi. Hal ini menandakan bahwa bilangan quaternion murni tidaklah tertutup. Hasil perkalian dari keduanya dapat dilihat di bawah ini:

$$q_1q_2 = [-(x_1x_2 + y_1y_2 + z_1z_2) + i(y_1z_2 - y_2z_1) + j(z_1x_2 - z_2x_1) + k(x_1y_2 - x_2y_1)]$$

Setelah itu, konjugasi dari bilangan quaternion dapat ditemukan dengan definisi dari konjugasi itu sendiri:

$$\overline{q_1} = s_1 - v_1 = s_1 - (ix_1 + jy_1 + kz_1)$$

Properti selanjutnya adalah balikan dari bilangan quaternion atau *inverse quaternion*. Simbol balikan bilangna quaterion q_1 adalah q_1^{-1} dengan persamaan balikan seperti di bawah ini:

$$q_1^{-1} = \frac{\overline{q_1}}{||q_1||^2}$$

Dimana persamaan diatas memenuhi persamaan berikut ini:

$$q_1q_1^{-1} = \frac{(s_1 - (ix_1 + jy_1 + kz_1))(s_1 + (ix_1 + jy_1 + kz_1))}{||q_1||^2} = 1$$

Balikan dari bilangan quaternion telah terbukti komutatif walaupun perkalian dari bilangan quaternion sendiri tidak.

Selanjutnya pada subab ini akan dibahas axioma tentang aljabar bilangan quaternion[2].

Diberikan

$$q, q_1, q_2, q_3 \in C$$

Closure

Untuk semua q_1 dan q_2

Penjumlahan $q_1 + q_2 \in C$

Perkalian $q_1q_2 \in C$

Identitas
Untuk tiap q, tedapat elemen identitas 0 dan 1 sehingga:

Penjumalahan q+0=0+q=q(0=0+i0+j0+k0)

Perkalian q(1) = (1)q = q(1=1+i0+j0+k0)

Balikan
Untuk tiap q terapat elemen balikan –q dan q^{-1} sehingga:

Penjumlaha q + (-q) = -q + q = 0

Perkalian $qq^{-1} = q^{-1}q = 1$ ($q \neq 0$)

Asosiatif
Untuk semua q_1, q_2, dan q_3

Penjumlahan $q_1 + (q_2 + q_3) = (q_1 + q_2) + q_3$

Perkalian $q_1q_2 \neq q_2q_1$

Distributif
Untuk semua q_1, q_2, dan q_3

$q_1(q_2 + q_3) = q_1q_2 + q_1q_3$

$(q_1 + q_2)q_3 = q_1q_3 + q_2q_3$

C. Rotasi Vektor Menggunakan Bilangan Quartenion

Pada subab ini akan dibahas rotasi vektor menggunakan bilangan quartenion. Sebenarnya rotasi vektor ini termasuk properti bilangan quartenion. Namun karena kaitan dengan fungsi SLERP yang menjadi fokus pembahasan makalah ini sangat besar, maka untuk properti ini dikhususkan sendiri subabnya.

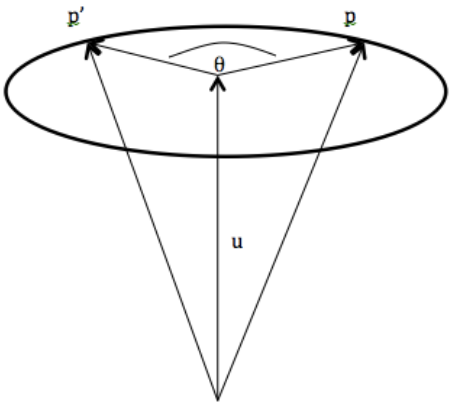
Sebenarnya rotasi vektor ini meruapakan salah satu apliaksi dari bilangan quaternion. Kita dapat memisalkan suatu vektor p akan dirotasikan sebanyak θ derajat pada sumbu axis yang diberikan oleh unit vektor p. Vektor p yang telah dirotasikan kita simbolkan dengan p' dan didapat dengan operasi seperti berikut[2]:

$$p' = qpq^{-1}$$

Dimana penjabaran tiap variabelnya adalah seperti di bawah ini:

$$p = xi + yj + zk$$
$$q = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\hat{u}$$
$$q^{-1} = \cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right)\hat{u}$$
$$\hat{u} = [x_u i + y_u j + z_u k]$$
$$||\hat{u}|| = 1$$

Ilustrasi dari rotasi vektor diatas dapat dilihat dibawah ini:



IV. PEMBAHASAN

A. SLERP

Sebenarnya pada *computer animation*, merotasikan sebuah karakter memiliki banyak alternatif cara yaitu menggunakan sudut Euler, rotasi matriks, ataupun yang kita bahas pada makalah ini, yaitu rotasi menggunakan bilangan quartenion. Namun selain cara merotasi yang bervariasi, ternyata fungsi untuk mengimplementasikan rotasi tersebut jugalah banyak, yaitu LERP, SLERP, dan NLERP. Penulis telah menyatakan di pendahuluan bahwa yang akan dibahas benar-benar pada makalah ini adalah SLERP. Namun, sebelum langsung membahas SLERP penulis akan menjelaskan sedikit perbedaan dari ketiga fungsi interpolasi ini.

Yang pertama adalah LERP yang memiliki kepanjangan *linear interpolation.* LERP merupakan formula geometri dasar dengan persamaan seperti di bawah ini[1]:

$$LERP(p_0, p_1; t) = (1 - t)p_0 + tp_1$$

Namun karena fungsi ini hanya dapat menghitung perpindahan sebuah vektor secara linear, maka implentasi ke bidang animasi menjadi tidak terlalu berguna. Hal ini dikarenakan dalam animasi, sebuah objek bergerak tidak hanya pada satu garis lurus, namun bergerak pada bidang tiga dimensi.

Oleh karena itu, SLERP yang memiliki kepanjangan spherical liner interpolation, memiliki kegunaan lebih dibandingkan LERP. SLERP membuat animasi rotasi terlihat lebih halus karena vektor tidak hanya bergerak pada garis lurus saja namun bergerak pada sebuah kurva. Maka dapat dikatakan bahwa SLERP merupakan LERP yang dilakukan pada bidang suatu bola.

Persamaan dari SLERP dapat dilihat di bawah ini:

$$SLERP(p_0, p_1; t) = \frac{\sin(1-t)\Omega}{sin\Omega}p_0 + \frac{\sin t\Omega}{sin\Omega}p_1$$
$$SLERP(q_1, q_2; t) = q_1(q_1^{-1}q_2)^t$$

Sedangkan NLERP adalah kepanjangan dari

normalized SLERP. Sehingga persamaan dari NLERP menjadi seperti dibawah ini:

$$NLERP(|q_1|,|q_2|;t) = |q_1|(|q_1|^{-1}|q_2|)^t$$

Keuntungan dari NLERP adalah bahwa persamaanya memiliki sifat komutatif, dimana sifat ini membuat fungsi tidak mudah menghasilkan suatu kegagalan. Selain itu NLERP sedikit lebih cepat digunakan dibandingkan SLERP. Namun tidak berarti NLERP lebih baik dibandingkan dengan SLERP. Sebuah objek yang dikomputasikan mengunakan fungsi NLERP cenderung bergerak lebih cepat diakhir dan diawal, sedangkan SLERP akan menghasilkan kecepatan konstan.

B. Fungsi SLERP dalam Bahasa C

Fungsi SLERP dapat dituliskan dalam sebuah kode pemograman yang dalam hal ini penulis memilih memperlihatkannya dalam bahasa C yang dikutip dari [4].

```
quat slerp(quat qa, quat qb, double t)
{
// quaternion yang akan di-return
quat qm
//kalkulasi sudut antara qa dan qb
double cosHalfTheta = qa.w * qb.w +
qa.x * qb.x + qa.y * qb.y + qa.z *
qb.z;
// jika qa=qb or qa=-qb maka theta = 0
bisa kita kembalikan qm=qa
if (abs(cosHalfTheta) >= 1.0){
      qm.w = qa.w;
      qm.x = qa.x;
      qm.y = qa.y;
      qm.z = qa.z;
return qm;
}
// Kalkulasi nilai sementara
double halfTheta = acos(cosHalfTheta);
double sinHalfTheta = sqrt(1.0 -
cosHalfTheta*cosHalfTheta);
// Jika theta = 180 derajat maka hasil
tidak sepenuhnya terdefinisi
// Kita dapat merotasi disekitar sumbu
axis normal ke qa atau qb
if (fabs(sinHalfTheta) < 0.001){
// fabs adalah floating point absolute
qm.w = (qa.w * 0.5 + qb.w * 0.5);
qm.x = (qa.x * 0.5 + qb.x * 0.5);
qm.y = (qa.y * 0.5 + qb.y * 0.5);
qm.z = (qa.z * 0.5 + qb.z * 0.5);
return qm;
}
double ratioA = sin((1 - t) *
halfTheta) / sinHalfTheta;
double ratioB = sin(t * halfTheta) /
sinHalfTheta;
//kalkulasi bilangan quaternion
qm.w = (qa.w * ratioA + qb.w *
ratioB);
qm.x = (qa.x * ratioA + qb.x *
ratioB);
qm.y = (qa.y * ratioA + qb.y *
ratioB);
qm.z = (qa.z * ratioA + qb.z *
ratioB);

return qm;
}
```

Kode diatas menghasilkan hasil tidak terdefinisi jika didapatkan theta sudutnya 180 derajat. Hal ini dikarenakan tidak ditemukannya lintasan terpendek untuk merotasi bilangan quaternion.

C. Interpolasi Bezier pada Bidang 4 Dimensi

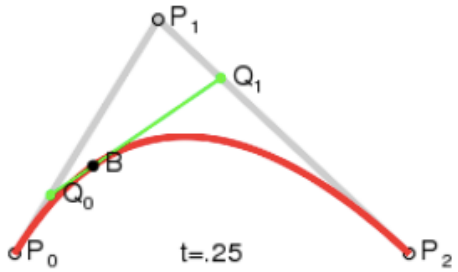
Fungsi SLERP dengan simple mengkalkulasi hanya dua buah bilangan quaternion. Namun, apa yang terjadi jika dibutuhkan hasil interpolasi lebih dari dua bilangan quaternion?

Misalnya kita asumsikan bahwa ada tiga bilangan quaternion yaitu q_{n-1}, q_n, dan q_{n+1}. Untuk mencari hasil interpolasi dari ketiga bilangan quaternion tersebut digunakan Kurva Bezier. Kurva Bezier adalah cara untuk mencegah pencarian hasil interpolasi yang menghasilkan kurva yang tidak bisa diturunkan (*non-differentiable*). Hal ini dikarenakan kurva Bezier selalu bersifat parametrik dan merupakan kurva yang memiliki keturunan (*differentiable*). Selain itu kurva Bezier untuk mencari hasil interpolasi merupakan aplikasi dari fungsi SLERP.

Maka dari itu, untuk mendapatkan kurva Bezier dari ketiga bilangan quaternion, harus dilakukan [3]:

1. Menambahkan dua bilangan quaternion, a_n dan b_n.
2. Menggambar kurva Bezier dengan parameter q_n, a_n, b_n, dan q_{n+1}.
3. Mengaplikasikan SLERP kepada bilangan quaternion di kurva, mengkalkulasi angka yang diinginkan diantara titik.
4. Mengkonversi data kembali ke format original.

Berikut adalah ilustrasi dari pembentukan kurva dua dimensi Bezier yang diambil dari referensi [3]:



Untuk membentuk bilangan quaternion a dan b kita dapat mengaprokimasi turunan dari tiga bilangan quaternion yang telah dimisalkan sebelumnya yaitu q_{n-1}, q_n, dan q_{n+1} dengan merata-ratakan titik pusat perbedaan antara ketiganya. Digunakan sebuah formula Shoemaker seperti dibawah ini[3]:

$$Biscet(p,q) = \frac{p+q}{|p+q|}$$
$$Double(p,q) = 2 * (p * q) * q - p$$
$$a_n = Biscet\left(Double\left(q_{n-1}, q_n\right), q_{n+1}\right)$$
$$b_n = Double\left(a_n, q_n\right)$$

Kemudian dilanjutkan dnegan membangun kurva Bezier bersama bilangan quaternion $q_{n,}$ $a_{n,}$ $b_{n,}$ dan $q_{n+1.:}$

$$P_{0,0} = q_n \quad P_{1,0} = a_n$$
$$P_{2,0} = b_{n+1} \quad P_{3,0} = q_{n+1}$$
$$P_{0,1} = SLERP(P_{0,0}, P_{1,0}, t)$$
$$P_{1,1} = SLERP(P_{1,0}, P_{2,0}, t)$$
$$P_{2,1} = SLERP(P_{2,0}, P_{3,0}, t)$$
$$P_{0,2} = SLERP(P_{0,1}, P_{1,1}, t)$$
$$P_{1,2} = SLERP(P_{1,1}, P_{2,1}, t)$$
$$P_{0,3} = SLERP\left(P_{0,2}, P_{2,1}, t\right) = q_{n+t}$$

V. KESIMPULAN

Dalam bidang animasi fungsi SLERP atau *spherical linear interpolation* dalam computer animation sangatlah membantu perkembangan industri animasi sampai sepesat sekarang. Fungsi SLERP merupakan salah satu dari tiga fungsi interpolasi pada computer animation, dimana dua lainnya adalah NLERP dan LERP. Perpindahan sebuah objek dihitung menggunakan apliaksi dari rotasi vektor menggunakan bilangan quaternion. Selain itu fungsi SLERP juga memiliki pengembangan contohnya kurva Beizer yang dapat menghitung perpindahan lebih dari dua bilangan quaternion.

REFERENSI

[1] http://web.mit.edu/2.998/www/QuaternionReport1.pdf Diakses tanggal 11 Des 2015 2:45 sore

[2] Vince, John, Geometric Algebra for Computer Graphic. London: Springer.

[3] http://embots.dfki.de/doc/seminar_ca/Kremer_Quaternions.pdf Diakses tanggal 13 Des 2015 19:37 WIB

[4] http://www.euclideanspace.com/maths/algebra/realNormedAlgebra/quaternions/slerp/ Diakses tanggal 14 Des 2015 9:32 WIB

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 27 November 2013

ttd