

Intersection of circles in 3D space

C. Scozzafava

southerlies.eu@gmail.com
www.southerlies.eu

Version 1.0

September 26, 2008

This document may be reproduced and distributed in whole or in part, in any medium physical or electronic, as long as this copyright notice is retained on all copies. Commercial redistribution is not allowed. All translations, derivative works, or aggregate works incorporating this document in whole or in part must be covered under this copyright notice. That is, you may not produce a derivative work from this document and impose additional restrictions on its distribution. For all not mentioned conditions please consider this document as released under the Creative Commons License. For further information please contact the author at southerlies.eu@gmail.com.

Questo documento può essere riprodotto e distribuito in tutto o in parte, con ogni mezzo fisico o elettronico, purché questo avviso di copyright sia mantenuto su tutte le copie. La ridistribuzione commerciale non é permessa. Ogni traduzione, lavoro derivato o comprendente questo documento deve contenere questo stesso avviso di copyright: per esempio, non si possono produrre lavori derivati da questo documento ed imporre restrizioni aggiuntive sulla sua distribuzione. Per tutte le condizioni non esplicitamente menzionate si prega di considerare questo documento come rilasciato sotto Licenza Creative Commons. Per ulteriori informazioni si prega di contattare l'autore all'indirizzo southerlies.eu@gmail.com.

Contents

1	Introduzione	4
1.1	Traslazioni in 2D	4
1.2	Estensione al caso 3D	4
1.2.1	Traslazione e Rotazione in 3D	5
2	Intersezione di circonferenze	5
2.1	Forma Canonica	6
3	Soluzione della Forma Canonica	6
3.1	Soluzione del caso complanare	7
3.2	Soluzione del caso sghembo	8
A	Appendice	10
A.1	Code Samples	10
A.1.1	Soluzione dell'equazione di secondo grado	10
A.1.2	Solving the circles intersection	11

Abstract

La rappresentazione parametrica delle curve è un potente e versatile strumento di modellazione per la rappresentazione software di questi tipi di oggetti matematici. Obiettivo di questa nota è la determinazione dei punti di intersezione fra due circonferenze comunque situate nello spazio 3D.

1 Introduzione

La rappresentazione parametrica più semplice di una circonferenza è quella, ben nota, che si riferisce al caso piano con centro nell'origine degli assi e raggio r . Detto P un generico punto dello spazio 2D si può scrivere che tale circonferenza, detta C_0 , è data da

$$C_0 = \left\{ P \in \mathbb{R}^2 : P = r \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix}, \theta \in [0, 2\pi[\right\}$$

1.1 Traslazioni in 2D

Supponiamo sia data una seconda circonferenza C_1 , anch'essa di raggio r , il cui centro si trovi nel punto $c_1 = [c_x \ c_y]^T$. Allora è molto semplice constatare che le due circonferenze C_0 e C_1 possono essere perfettamente sovrapposte tramite una traslazione della prima sulla seconda. L'entità di questa traslazione, misurata con un vettore, è pari alla differenza fra i due centri; ovvero $C = c_1 - c_0 = c_1 - 0 = c_1$.

Quindi, nel piano, una generica circonferenza C_1 avrà equazione parametrica pari a

$$C_1 = \left\{ P \in \mathbb{R}^2 : P = C + r \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix}, \theta \in [0, 2\pi[\right\} \quad (1)$$

1.2 Estensione al caso 3D

Come primo passo osserviamo come può essere descritta la C_0 nello spazio 3D. In questo caso notiamo che un punto P è definito da una tripletta di coordinate invece che da una coppia. La terza coordinata è riferita alla dimensione z . Nel caso della C_0 il valore di z sarà zero. La corrispondente equazione parametrica diventerà

$$C_0 = \left\{ P \in \mathbb{R}^3 : P = r \begin{bmatrix} \cos \theta \\ \sin \theta \\ 0 \end{bmatrix}, \theta \in [0, 2\pi[\right\}$$

1.2.1 Traslazione e Rotazione in 3D

Nello spazio a 3 dimensioni le trasformazioni che è possibile compiere sulla C_0 non sono solo traslazioni ma anche rotazioni intorno ad un asse passante per il centro. Queste ultime trasportano la circonferenza *fuori* dal piano $z = 0$.

Ruotare una circonferenza vuol dire ruotare tutti i punti che la compongono. Detta R una generica matrice di rotazione la circonferenza ruotata di R avrà una equazione della forma

$$C_{0,R} = \left\{ P \in \mathbb{R}^3 : P = rR \begin{bmatrix} \cos \theta \\ \sin \theta \\ 0 \end{bmatrix}, \theta \in [0, 2\pi[\right\}.$$

Tuttavia una generica circonferenza in 3D può essere caratterizzata oltre che da una rotazione rispetto agli assi coordinati anche da una certa traslazione rispetto all'origine. Quindi una equazione parametrica generica per la circonferenza in 3D è data da

$$C_{1,R} = \left\{ P \in \mathbb{R}^3 : P = C + rR \begin{bmatrix} \cos \theta \\ \sin \theta \\ 0 \end{bmatrix}, \theta \in [0, 2\pi[\right\} \quad (2)$$

2 Intersezione di circonferenze

Ci proponiamo ora, utilizzando il risultato ottenuto in (2), di calcolare i punti di intersezione, se esistono, di due circonferenze comunque posizionate (e orientate) nello spazio.

Siano date le circonferenze

$$C_a = C_a + r_a R_a \begin{bmatrix} \cos \theta \\ \sin \theta \\ 0 \end{bmatrix} \quad (3)$$

$$C_b = C_b + r_b R_b \begin{bmatrix} \cos \epsilon \\ \sin \epsilon \\ 0 \end{bmatrix} \quad (4)$$

Condizione necessaria affinché esistano dei punti di intersezione fra C_a e C_b è che esse abbiano punti in comune, ovvero che i punti di intersezione, se esistono, soddisfino entrambe le equazioni; ovvero che esistano dei valori di θ ed ϵ tali che risulti

$$C_a + r_a R_a \begin{bmatrix} \cos \theta \\ \sin \theta \\ 0 \end{bmatrix} = C_b + r_b R_b \begin{bmatrix} \cos \epsilon \\ \sin \epsilon \\ 0 \end{bmatrix} \quad (5)$$

2.1 Forma Canonica

L'equazione (5) può essere manipolata per ottenere

$$\begin{bmatrix} \cos \theta \\ \sin \theta \\ 0 \end{bmatrix} = \frac{r_b}{r_a} R_a^{-1} (C_b - C_a) + \frac{r_b}{r_a} R_a^{-1} R_b \begin{bmatrix} \cos \epsilon \\ \sin \epsilon \\ 0 \end{bmatrix}$$

che con le posizioni $b = \frac{r_b}{r_a} R_a^{-1} (C_b - C_a)$

e

$$R = \frac{r_b}{r_a} R_a^{-1} R_b$$

può essere riscritta¹ e riordinata come

$$\begin{bmatrix} \cos \theta \\ \sin \theta \\ 0 \end{bmatrix} = R \begin{bmatrix} \cos \epsilon \\ \sin \epsilon \\ 0 \end{bmatrix} + b \quad (6)$$

cui ci si riferirà nel seguito come alla *forma canonica*.

Proprietà della matrice R La matrice R che compare nell'equazione canonica (6) è stata definita come il prodotto fra due matrici di rotazione e lo scalare $\frac{r_b}{r_a}$. Questo vuol dire che per essa, pur non essendo una *vera* matrice di rotazione, sono valide le seguenti affermazioni:

- I vettori colonna(riga) di R sono mutuamente ortogonali.
- La norma di ogni vettore colonna(riga) è pari allo scalare $\frac{r_b}{r_a}$.
- Il determinante di R è pari allo scalare $\frac{r_b}{r_a}$.

3 Soluzione della Forma Canonica

La condizione necessaria per l'intersezione di due circonferenze, data in forma canonica, può essere espansa nel seguente sistema di equazioni

$$\cos \theta = r_{11} \cos \epsilon + r_{12} \sin \epsilon + b_1 \quad (7a)$$

$$\sin \theta = r_{21} \cos \epsilon + r_{22} \sin \epsilon + b_2 \quad (7b)$$

$$0 = r_{31} \cos \epsilon + r_{32} \sin \epsilon + b_3 \quad (7c)$$

¹La matrice R è comunque considerata di rotazione benchè sia stata moltiplicata per lo scalare non unitario $\frac{r_b}{r_a}$. Tuttavia questa posizione non lede nessuno dei risultati che saranno calcolati nel seguito.

La soluzione di questo sistema sarà affrontata considerando due possibili condizioni:

- Circonferenze complanari \rightarrow cui corrisponde una matrice di rotazione R la cui terza colonna è il versore canonico \hat{z} .
- Circonferenze sghembe \rightarrow cui corrisponde una matrice di rotazione generica.

3.1 Soluzione del caso complanare

Nel caso in cui le circonferenze giacciono nello stesso piano la terza delle (7a) si riduce all'identità

$$0 = b_3$$

Essa è, perciò, inutile ai fini della soluzione dei punti di intersezione, restando tuttavia un indicatore della validità numerica dei dati in ingresso: nel caso in cui tale identità non sia verificata è possibile affermare che i dati in ingresso al problema sono mal condizionati. Le restanti prime due equazioni delle (7a) sono quelle su cui concentreremo la nostra attenzione.

Il primo passo da eseguire consiste nell'eliminazione delle incognite nella variabile θ . Ciò si ottiene considerando l'equazione fondamentale della trigonometria

$$\cos^2 \alpha + \sin^2 \alpha = 1 \quad (8)$$

Quindi quadrando le due equazioni si ottiene

$$\begin{aligned} \cos^2 \theta &= r_{11}^2 \cos^2 \epsilon + r_{12}^2 \sin^2 \epsilon + b_1^2 + 2(r_{11}r_{12} \cos \epsilon \sin \epsilon + r_{11}b_1 \cos \epsilon + r_{12}b_1 \sin \epsilon) \\ \sin^2 \theta &= r_{21}^2 \cos^2 \epsilon + r_{22}^2 \sin^2 \epsilon + b_2^2 + 2(r_{21}r_{22} \cos \epsilon \sin \epsilon + r_{21}b_2 \cos \epsilon + r_{22}b_2 \sin \epsilon) \end{aligned}$$

dalla precedente e tenendo conto del fatto che (vedi 2.1):

- $r_{11}^2 + r_{21}^2 = r_{12}^2 + r_{22}^2 = \left(\frac{r_b}{r_a}\right)^2$
- $r_{11}r_{12} + r_{21}r_{22} = 0$

risulta

$$1 = \left(\frac{r_b}{r_a}\right)^2 + b_1^2 + b_2^2 + 2(r_{11}b_1 + r_{21}b_2) \cos \epsilon + 2(r_{12}b_1 + r_{22}b_2) \sin \epsilon \quad (9)$$

A questo punto utilizzando la (8) nella variabile ϵ è possibile trasformare la (9) in una equazione di secondo grado nell'incognita $\cos \epsilon$.

Una discussione delle soluzioni di questa equazione discriminerà sull'esistenza (soluzioni reali) o meno dei punti di intersezione e sulla loro eventuale degenerazione in un unico punto geometrico (soluzioni coincidenti).

Nel caso di esistenza di tali soluzioni, ovvero noti i valori di $\cos \epsilon$, si utilizzerà la (9) per il calcolo dei corrispondenti valori di $\sin \epsilon$.

I punti di intersezione, note le coppie seno-coseno potranno essere calcolati utilizzando l'equazione parametrica (4).

3.2 Soluzione del caso sghembo

Diversamente dal caso complanare sarà possibile utilizzare la terza delle (7a)² in associazione con la solita (8).

Analogamente a quanto illustrato nel caso precedente ci si potrà ancora ricondurre alla soluzione di un'equazione di secondo grado in $\cos \epsilon$ o $\sin \epsilon$.

²In questo caso questa equazione non sarà mai degenerare

References

- [1] D.Goldberg *What Every Computer Scientist Should Know About Floating-Point Arithmetic*, Computing Survey, 1991.

A Appendice

A.1 Code Samples

Vengono riportati degli estratti del codice utilizzato per implementare la soluzione descritta. Il linguaggio utilizzato è il C#.

A.1.1 Soluzione dell'equazione di secondo grado

Ai fini del calcolo dei punti di intersezione è necessaria la soluzione di un'equazione di secondo grado quando le radici siano reali. Il caso di radici complesse implica che il problema di partenza non abbia soluzione.

```
public static bool SolveSecondOrderReal
(
    double a,          //Quadratic coefficient
    double b,          //Linear coefficient
    double c,          //Known term
    double tolerance, //Tolerance to check the values against zero
    ref double s1,     //First real root
    ref double s2      //Second real root
)
{
    if (Math.Abs(a) < tolerance)
    {
        if (SolveFirstOrder(b, c, tolerance, ref s1) == false)
            return false;
        s2 = s1;
        return true;
    }

    double delta = b * b - 4.0 * a * c;
    double den   = 2.0 * a;

    if (delta < 0.0)
        return false;

    delta = Math.Sqrt(delta);

    //Solve roots trying to avoid 'catastrophic' cancellation
    if (b < 0.0)
    {
        // Original phormula
        //s1 = (-b + delta) / den;
        //s2 = 2.0 * c / (-b + delta);

        //dummy to avoid twice the same sum!
```

```

        s2 = - b + delta;

        s1 = s2 / den;
        s2 = 2.0 * c / s2;
    }
    else
    {
        // Original phormula
        //s1 = -2.0 * c / (b + delta);
        //s2 = (-b - delta) / den;

        //dummy to avoid twice the same sum!
        s2 = - b - delta;

        s1 = 2.0 * c / s2;
        s2 = s2 / den;
    }

    return true;
} //SolveSecondOrderReal

```

A.1.2 Solving the circles intersection

```

public static bool Intersection
(
    Vector3 p,           //Center of the first circle
    double rp,          //Radius of first circle
    Matrix3 nu,         //Attitude of first circle
    Vector3 q,          //Center of the second circle
    double rq,          //Radius of second circle
    Matrix3 mu,         //Attitude of second circle
    double tolerance,  //Tolerance to check the values against zero
    ref List<Vector3> result //Intersection points, if any.
)
{
    Vector3 b;
    Matrix3 tmu;
    double c2_1, c2_2, s2_1, s2_2;
    double c1_1, c1_2, s1_1, s1_2;
    c2_1 = c2_2 = s2_1 = s2_2 = 2.0;

    //reset solutions
    result.Clear();

    //known term
    b = Vector3.Subtract(q, p);

```

```

b.StretchBy(1.0 / rp);
b.TransformByInverse(nu);

//matrix term
tmu = mu.LeftProductByInverse(nu);
tmu.StretchBy(rq / rp);

//First check : if z versor of mu is parallel to z world
if (Math.Abs(Math.Abs(tmu.M[2, 2]) - 1.0) < tolerance)
{
    //same plane!
    if (Math.Abs(b.Z) > tolerance)
        return false;//data error

    //find intersection on plane
    double t0, t1, t2;

    c2_1 = (rq / rp) * (rq / rp);

    t0 = b.X * tmu.M[0, 0] + b.Y * tmu.M[1, 0];
    t1 = b.X * tmu.M[0, 1] + b.Y * tmu.M[1, 1];
    t2 = (b.X * b.X + b.Y * b.Y + c2_1 - 1.0) / 2.0;

    c2_1 = t1 * t1;//used to avoid twice a product

    /*
    al = t0 * t0 + t1 * t1;
    be = 2.0 * t0 * t2;
    ga = t2 * t2 - t1 * t1;
    */

    if (PolynomialEquation.SolveSecondOrderReal(t0 * t0 + c2_1,
                                                2.0 * t0 * t2,
                                                t2 * t2 - c2_1,
                                                tolerance,
                                                ref c2_1,
                                                ref c2_2) == false)

        return false;

    if (Math.Abs(c2_1) <= 1.0)
    {
        if (PolynomialEquation.SolveFirstOrder(t1,
                                                t0 * c2_1 + t2,
                                                0.001,
                                                ref s2_1) == false)
            return false;
    }
}

```

```

}

if (Math.Abs(c2_2) <= 1.0)
{
    if (PolynomialEquation.SolveFirstOrder(t1,
        t0 * c2_2 + t2,
        0.001,
        ref s2_2) == false)
        return false;
}

//Evaluates cos(theta) and sin(theta)
c1_1 = tmu.M[0, 0] * c2_1 + tmu.M[0, 1] * s2_1 + b.X;
s1_1 = tmu.M[1, 0] * c2_1 + tmu.M[1, 1] * s2_1 + b.Y;

c1_2 = tmu.M[0, 0] * c2_2 + tmu.M[0, 1] * s2_2 + b.X;
s1_2 = tmu.M[1, 0] * c2_2 + tmu.M[1, 1] * s2_2 + b.Y;
}
else
{
    if ((Math.Abs(tmu.M[2, 0]) + Math.Abs(tmu.M[2, 1])) < tolerance)
    {
        //data error
        return false;
    }

    //Use third row and the trigonometric equation to
    // solve cos and sin for second circle
    /*
    a = tmu.M[2,0]*tmu.M[2,0] + tmu.M[2,1]*tmu.M[2,1];
    b = 2 * b.Z * tmu.M[2,0];
    c = b.Z*b.Z - tmu.M[2,1]*tmu.M[2,1];
    */
    c2_1 = tmu.M[2, 1] * tmu.M[2, 1]; //used to avoid a product
    if (PolynomialEquation.SolveSecondOrderReal
        (tmu.M[2, 0] * tmu.M[2, 0] + c2_1/*tmu.M[2, 1] * tmu.M[2, 1]*/,
        2 * b.Z * tmu.M[2, 0],
        b.Z * b.Z - c2_1/*tmu.M[2, 1] * tmu.M[2, 1]*/,
        tolerance,
        ref c2_1,
        ref c2_2) == false)
        return false;

    if (Math.Abs(c2_1) <= 1.0)
    {
        if (PolynomialEquation.SolveFirstOrder(tmu.M[2, 1],

```

```

    tmu.M[2, 0] * c2_1 + b.Z,
    0.001,
    ref s2_1) == false)
    return false;
}

if (Math.Abs(c2_2) <= 1.0)
{
    if (PolynomialEquation.SolveFirstOrder(tmu.M[2, 1],
        tmu.M[2, 0] * c2_2 + b.Z,
        0.001,
        ref s2_2) == false)
        return false;
}

//Evaluates cos(theta) and sin(theta)
c1_1 = tmu.M[0, 0] * c2_1 + tmu.M[0, 1] * s2_1 + b.X;
s1_1 = tmu.M[1, 0] * c2_1 + tmu.M[1, 1] * s2_1 + b.Y;

c1_2 = tmu.M[0, 0] * c2_2 + tmu.M[0, 1] * s2_2 + b.X;
s1_2 = tmu.M[1, 0] * c2_2 + tmu.M[1, 1] * s2_2 + b.Y;
}

//to solve points using first circle
Vector3 inters_01 = new Vector3(rp * c1_1, rp * s1_1, 0.0);
inters_01.Transform(nu);
inters_01.Add(p);

Vector3 inters_02 = new Vector3(rp * c1_2, rp * s1_2, 0.0);
inters_02.Transform(nu);
inters_02.Add(p);

result.Add(inters_01);
result.Add(inters_02);
return true;
} //Intersection

```