# Revisión del algoritmo GJK para la detección de colisiones en modelos poliédricos convexos

Claudia MARMOLEJO RIVAS[1, 2], Francisco MARMOLEJO RIVAS[3], Juan Manuel IBARRA ZANNATHA[2]

1Laboratorio de Robótica
Depto. de Control Automático
del CINVESTAV
Av. IPN No. 2508, Lindavista,
07300 México, DF
jibarra@ctrl.cinvestav.mx

2Escuela de Informática, U.
Autónoma de Sinaloa
Av. Universidad y Av.
Leonismo Internacional s/n
82017 Mazatlán, Sin.
cmarmolejo@ctrl.cinvestav.mx

3Instituto de Matemáticas,
UNAM
Área de la Inv. Científica
Ciudad Universitaria
04510 México D.F:
quico@math.unam.mx

## Resumen

*En este trabajo se estudia el algoritmo GJK para la detección de colisiones por su factibilidad de implementación en la simulación de dos o más cuerpos virtuales que entrarán en contacto, se ilustra su aplicación con ejemplos numéricos.*

## 1. Introducción

Cuando en una escena virtual se representan varios objetos animados existe la posibilidad de que entren en contacto, por lo que es necesario, primero detectar la colisión y segundo responder a ella, lo primero es un problema cinemático que involucra la posición relativa de los objetos en el mundo virtual, lo segundo es un problema dinámico en el que es necesario predecir el comportamiento acorde con las leyes físicas apropiadas [7]. La Geometría computacional plantea el problema en términos de un ambiente estático, mientras que la Robótica lo hace desde el punto de vista dinámico, puesto que la simulación dinámica se resuelve mediante una secuencia de problemas estáticos, uno por intervalo de tiempo, es posible aplicar el mismo algoritmo en los dos casos [1].

Hay dos tipos de algoritmos de detección de colisiones en modelos poliédricos convexos representados por la especificación de sus fronteras: los basados en rasgos y los basados en simplejos, los rasgos son los vértices, las aristas y las caras que forman sus fronteras, los algoritmos basados en rasgos ejecutan cálculos geométricos en sus elementos para saber si son disjuntos, calculan la distancia mínima entre el par de politopos convexos, cuando la distancia entre ellos es cero, el par de poliedros convexos están en contacto [5, 6]. Las aproximaciones que usa la programación lineal se basan en la existencia de un plano de separación entre la pareja de poliedros convexos. En este trabajo se revisa un método para la detección de colisiones basado en simplejos pues su implementación es factible [8] debido a su sencillez.

## 2. Simplejos

Puntos, aristas, triángulos y tetraedros son ejemplos de simplejos de baja dimensión.

Se usan combinaciones de puntos para definir simplejos en cualquier dimensión. Sea $S$ un conjunto finito $d$-dimensional en el espacio Euclidiano denotado como $R^d$, Se denomina a $S$ conjunto afín si se cumple para cada $x$ y $y$ en $S$ que $(1-\lambda)x + \lambda y \in S$ y $\lambda \in R$. El conjunto vacío y el espacio $R^d$ son ejemplos extremos de conjuntos afines, La definición también comprende a un único punto, en general, un conjunto afín debe contener, además de a cualesquiera dos puntos de $S$, a toda la línea que los une. La imagen intuitiva es la de una estructura no curvada, como una línea o un plano en el espacio [9]. Una combinación afín de puntos $p_i \in S$ es un punto $x = \alpha_i p_i$ con $\sum \alpha_i = 1$. El casco afín, **aff** $S$, es el conjunto de todas las combinaciones afines, es decir, es la intersección de todos los hiperplanos que contienen a $S$. Los puntos en $S$ son afínmente independientes (a.i.) si ninguno es la combinación de los otros puntos en $S$. Por ejemplo, el casco afín de tres puntos a.i. es un plano, el de dos puntos a.i. es una recta y el casco convexo de un único punto es el punto mismo. Una combinación convexa es una combinación afín con coeficientes no-negativos: $\alpha_i \geq 0$ para todo $p_i$ en $S$. El casco convexo, **conv** $S$, es el conjunto de todas las combinaciones convexas, es decir, es la combinación de todos los semiplanos que contienen a $S$. Un simplejo es el casco convexo del conjunto de puntos a.i. Si $S \subseteq R^d$ es el conjunto de $k+1$ puntos a.i., entonces la dimensión del simplex $\sigma$ = **conv** $S$ es dim $\sigma = k$ y $\sigma$ se denomina *k-simplex*. El número más grande de puntos afínmente independientes en $R^d$ es $d+1$, y se tienen simplejos de dimensión -1 en $d$. Hay cinco tipos de simplejos en $R^3$: el conjunto vacío es de dimensión -1, el punto es de dimensión 0, la recta que une 2 puntos es de dimensión 1, el triángulo formado por 3 puntos es de dimensión 2 y el tetraedro que consta de 4 vértices es de dimensión 3 [2].

## 3. Algoritmo GJK [3]

El algoritmo diseñado por Gilbert, Johnson y Keerthi (GJK) constituye uno de los primeros

ejemplos de algoritmos basados en simplejos, es un método iterativo que calcula la distancia mínima entre dos objetos convexos *A* y *B*. En cada iteración se construye un simplejo contenido en *A* – *B* que se halla más cerca del origen que aquel generado en la iteración anterior. El conjunto de vértices del simplejo construido en la k-ésima iteración se denota mediante $V_k$, mientras que el punto de este simplejo más cercano al origen es representado por $v_k$, pues dados dos poliedros *A* y *B*, la distancia más corta al origen del simplejo definido por los vértices del poliedro resultante de la diferencia de Minkowski (*A* – *B*) corresponde con la distancia más corta existente entre ambos poliedros (**Fig**. 1), si el origen está dentro del poliedro resultante, entonces los poliedros originales de interpenetran y está disponible una estimación de la medida de la interpenetración.

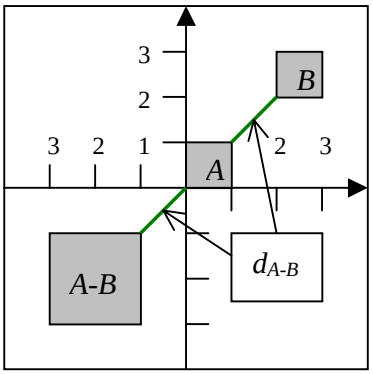


**Fig. 1**: El cálculo de la distancia entre 2 politopos se hace hallando el punto más cercano al origen de un tercer politopo convexo generado a partir de los otros dos

## 3.1 Definiciones

Dados dos objetos *A* y *B* en el espacio tridimensional conviene representarlos como conjuntos compactos $K_A$, $K_B \subset R^3$, La distancia Euclidiana entre los objetos *A* y *B* está definida por el par de puntos más cercanos entre $K_A$ y $K_B$:

$$d_{(k)} = \min\{|z| : z \in K \wedge K = K_A - K_B\}. \quad (1.1)$$

El conjunto diferencia de Minkowski se denota por:

$$K = \{a - b : a \in K_A \wedge b \in K_B\}. \quad (1.2)$$

La envolvente afín por:

$$aff\ K = \left\{\sum_{i=1}^{l} \lambda_i x_i : x_i \in K, \lambda_1 + \ldots + \lambda_l = 1\right\} \quad (1.3)$$

El casco convexo mediante:

$$co\ K = \left\{\sum_{i=1}^{l} \lambda_i x_i : x_i \in K, \lambda_i \geq 0, \lambda_1 + \ldots + \lambda_l = 1\right\}$$

El punto de *K* más cercano al origen *v*(*K*) será:

$$v(K) \in K, |v(K)| = \min\{|x| : x \in K\}. \quad (1.4)$$

cuya representación tiene la siguiente forma:

$$v(co\ K) = \sum_{i=1}^{l} \lambda_i x_i \in K, \quad (1.5)$$

$$\lambda_i > 0, \lambda_1 + \ldots + \lambda_l = 1 l \leq \dim K + 1$$

En $R^m$, un hyperplano *h* se define por: $h = \{x \in R^m : x \square \eta = \beta\}$ donde $\eta \neq 0$ es normal a *h*. *h* es un subespacio afín de $R^m$ de dimensión *d*–1. *h* divide a $R^m$ en dos semiespacios que pueden ser abiertos o cerrados, el par de semiespacios cerrados de *h* se definen como $h- = \{x \in R^m : x \square \eta \leq \beta\}$ y $h+ = \{x \in R^m : x \square \eta \geq \beta\}$.

Si *K* es un conjunto convexo compacto, $K \cup h \neq \varnothing$ y *K* esta en uno de los semiespacios cerrados definidos por *h*, entonces *h* es un plano soporte de *K*. La función soporte $h_K : R^m \rightarrow R$ se define por:

$$h_K(\eta) = \max\{x \square \eta : x \in K\}. \quad (1.6)$$

La solución de contacto se denota por $s_K(\eta) : R^m \rightarrow K$ y constituye una solución a (1.6) donde $s_K(\eta)$ satisface:

$$h_K(\eta) = s_K(\eta) \square \eta, s_K(\eta) \in K\ \forall \eta \in R^m \quad (1.7)$$

$s_K(\eta)$ define el punto más lejano de *K* en la dirección de $\eta$. Es posible definir el conjunto diferencia $K = K_A - K_B$ en términos de sus funciones soporte y de contacto mediante:

$$h_K(\eta) = h_{K_A}(\eta) + h_{K_B}(-\eta)$$

$$s_K(\eta) = s_{K_A}(\eta) - s_{K_A}(-\eta)$$

## 3.2 El Algoritmo

*Teorema* 1: Sea $K \subset R^m$ compacto y convexo, se define $g_K(x) : R^m \rightarrow R$ como:

$$g_K(x) = |x|^2 + h_K(-x) \quad (1.8)$$

para $x \in K$, entonces

1. si $g_K(x) > 0$ existe un punto *z* en el segmento de línea $\{x, s_K(-x)\}$ que satisface $|z| < |x|$;
2. $x = v(K)$ si y sólo si $g_K(x) = 0$
3. $|x - v(K)|^2 \leq g_K(x)$.

*Demostración* del teorema 1:

1. Suponiendo que $|s_K(-x)| \geq |x|$. Se definen:

$$z = x + \lambda(s_K(-x) - x),\ 0 \leq \lambda \leq 1$$

$$\lambda = g_K(x)/|s_K(-x) - x|^2, \quad (1.9)$$

vemos que:

$$|x - s_K(-x)|^2 = |x|^2 - 2(x \square s_K(-x)) + |s_K(-x)|^2$$

$$|x - s_K(-x)|^2 \geq 2(|x|^2 + h_K(-x))$$

$$|x - s_K(-x)|^2 \geq 2 g_K(x)$$

Sustituyendo este resultado en la ecuación (1.9) se tiene que: $0 \geq \lambda \geq 1/2$, luego

$$z \in co\{x, s_K(-x)\},$$

$|z|^2 = |x + \lambda(s_K(-x) - x)|^2 \leq$

$|x|^2 - \lambda|x - s_K(-x)|^2 =$

$|x|^2 - g_K(x) < |x|^2$.

2a. Sea $g_K(x) = |x|^2 + h_K(-x) = 0$,

$|x|^2 = -h_K(-x) = \min\{\omega\cdot(-x) : \omega \in K\}$

se puede escribir que:

$|x|^2 \leq |x|^2 + |\omega - x|^2 = |\omega|^2 + 2(|x|^2 - x\cdot\omega) =$

$|\omega|^2 + 2(|x|^2 + h_K(-x)) = |\omega|^2 \ \forall \omega \in K,$

luego $x = v(K)$.

2b. Sea $x = v(K)$, si $g_K(x) > 0$, $x \neq v(K)$ entonces $g_K(x) \leq 0$ pero como $x \in K$ $g_K(x) \geq 0$ se concluye que $g_K(x) = 0$.

3) El resultado obtenido en 2) implica que $|v(K)|^2 = -h_K(-v(K)) \leq \omega\cdot v(K) \ \forall \omega \in K$,

en consecuencia ,

$|x - v(K)|^2 \leq |x|^2 - x\cdot v(K) = |x|^2 + h_K(-x) = g_K(x)$

*Algoritmo para calcular la distancia*

Dado un conjunto convexo compacto $K \subset R^m$ y puntos iniciales $y_1,\ldots,y_v \in K$, $1 \leq v \leq m+1$, ejecutar los siguientes pasos:

1. hacer $V_0 = \{y_1, \ldots, y_v\}$ y $k=0$;
2. calcular $v_k = v(\text{co } V_k)$;
3. si $g_K(v_k) = 0$, hacer $g_K(v_k) = 0$, y detenerse
4. hacer $V_{k+1} = \hat{V}_k \cap \{s_K(-v_k)\}$, donde, $\hat{V}_k \subset V_k$ tiene $m$ elementos o menos y satisface $v_k \in co\,\hat{V}_k$, incrementar $k$ y volver al paso 2.

El algoritmo para calcular la distancia requiere, en el paso 2., que se halle $v(co\,Y_s), Y_s = \{y_1,\ldots,y_v\} \subset R^m$ donde $Y_s = V_k$ con:

$$v(co\,V_k) = \sum_{i\in I_s} \lambda_i y_i, \ \sum_{i\in I_s} \lambda_i = 1, \ \lambda_i > 0, \qquad (1.10)$$

$I_s \in \{1,\ldots,v\}$ and $V_k = Y_s = \{y_i : i \in I_s\}$ a.i.

donde $s$ indica un miembro particular de la familia de todos los subconjuntos no vacíos de $Y$. Observe que $Y_s$ se convierte en $V_k$ en el paso 4 del algoritmo de la distancia. El número de subconjuntos de $Y$ se pueden calcular mediante:

$$\sigma = \sum_{k=1}^{v} \left[ v!/k!(v-k)! \right]$$

Geométricamente esto significa hallar para cada uno de los politopos del *co* $Y_s$, $v_k$ y ver si contienen *v(co Y)*. Sí $v = 3$, entonces $m = 2$ y hay a lo más $\sigma = 7$ subconjuntos a examinar, el *co* $Y_s$ es un simplex tridimensional donde los 7 subconjuntos son: 3 vértices, 3 aristas y 1 triángulo. Sean $I'_s$ el complemento de $I_s$ en $I$, $Y_s$, $s = 1,\ldots,\sigma$ un ordenamiento de los subconjuntos de $Y$. Se definen los siguientes números reales: $\Delta_i(Y_s)$, $i \in I_s$ y $\Delta(Y_s)$ por:

$$\Delta_i(\{y_i\}) = 1, i \in I$$

$$\Delta_j(Y_s \cup \{y_j\}) = \sum \Delta_i(Y_s)(y_i\cdot y_k - y_i\cdot y_j), k \in I_s, \ j \in I'_s$$

$$\Delta(Y_s) = \sum_{i\in I_s} \Delta_i(Y_s). \qquad (1.11)$$

*Teorema* 2: La representación (1.10) es verdadera si y sólo si:

1. $\Delta(Y_s) > 0$ que es la condición de independencia afín.;
2. $\Delta_i(Y_s) > 0 \ \forall \ i \in I_s$ asegura que $v(co\,Y_s) \in Y_s$;
3. $\Delta_j(Y_s \cap \{y_j\}) \leq 0$ para cada $j \in I_s$ $\Delta_j(Y_s \cup \{y_j\}) \leq 0 \ \forall j \in I'_s$ implica que $v(co\,Y_s) = v(co\,Y)$ con $\lambda_i = \Delta_i(Y_s)/\Delta(Y_s)$ (1.12)

*Demostración* del teorema 2:

Primero se considera el caso del cálculo de $v(aff\ Y_s)$, $Y_s \subset Y$. Sea $Y_s$ un subconjunto ordenado de $Y$ con $r > 1$ elementos representados por $x_1, \ldots, x_r$, en este caso:

$v(aff\ Y_s) = \sum_{i=1}^{r} \lambda_i x_i$ con $\sum_{i=1}^{r} \lambda_i = 1$ de donde

$\lambda_1 = 1 - \sum_{i=2}^{r} \lambda_i$ entonces se puede escribir

$$|v(aff\ Y_s)|^2 = f(\lambda_2,\ldots,\lambda_r) = \left| x_1 + \sum_{i=2}^{r} \lambda_i (x_i - x_1) \right|^2$$

Hallar la distancia más corta del politopo afínmente independiente $Y_s$ al origen, supone la obtención de $\lambda_2,\ldots,\lambda_r \in R$ tales que minimicen $f(\lambda_2,\ldots,\lambda_r)$, esta función convexa representa las distancias de los puntos en el espacio afín al origen, de manera que el mínimo debe ser un único punto, ello se logra mediante:

$\partial f(\lambda_2,\ldots,\lambda_r)/\partial\lambda_i = 0, \quad i = 2,\ldots,r$

$$f(\lambda_2,\ldots,\lambda_r) = \left| x_1 + \sum_{i=2}^{r} \lambda_i (x_i - x_1) \right|^2$$

Pero cada punto tiene $d$ componentes y:

$$f(\lambda_2,\ldots,\lambda_r) =$$

$$= \left( x_{11} + \sum_{i=2}^{r} \lambda_i (x_{i1} - x_{11}), \ldots, x_{1d} + \sum_{i=2}^{r} \lambda_i (x_{id} - x_{1d}) \right)^2$$

$$= \sum_{j=1}^{d} \left( x_{1j} + \sum_{i=1}^{r} \lambda_i (x_{ij} - x_{1j}) \right)^2$$

La derivada parcial de $f(\lambda_2,\ldots,\lambda_r)$ con respecto a $\lambda$ será entonces: $\partial f(\lambda_2,\ldots,\lambda_r)/\partial\lambda_k = 0$, es decir,

$$\sum_{j=1}^{d} 2\left( x_{1j} + \sum_{i=2}^{r} \lambda_i (x_{ij} - x_{1j}) \right)(x_{kj} - x_{1j}) = 0$$

que es igual a:

$$\sum_{j=1}^{d}\left(x_{1j}+\sum_{i=2}^{r}\lambda_i\left(x_{ij}-x_{1j}\right)\right)\left(x_{kj}-x_{1j}\right)=$$

$$\sum_{j=1}^{d}x_{1j}\left(x_{kj}-x_{1j}\right)+\sum_{j=1}^{d}\sum_{i=2}^{r}\lambda_i\left(x_{ij}-x_{1j}\right)\left(x_{kj}-x_{1j}\right)=$$

$$\sum_{j=1}^{d}x_{1j}\left(x_{kj}-x_{1j}\right)+\sum_{i=2}^{r}\sum_{j=1}^{d}\lambda_i\left(x_{ij}-x_{1j}\right)\left(x_{kj}-x_{1j}\right)=$$

$$x_1\cdot\left(x_k-x_1\right)+\sum_{i=2}^{r}\lambda_i\left(x_i-x_1\right)\cdot\left(x_k-x_1\right)=$$

$$x_1\cdot\left(x_k-x_1\right)+\sum_{i=2}^{r}\lambda_i x_i\cdot\left(x_k-x_1\right)-\sum_{i=2}^{r}\lambda_i x_1\cdot\left(x_k-x_1\right)=$$

$$x_1\cdot\left(x_k-x_1\right)+\sum_{i=2}^{r}\lambda_i x_i\cdot\left(x_k-x_1\right)-x_1\cdot\left(x_k-x_1\right)\sum_{i=2}^{r}\lambda_i=$$

$$\left(1-\sum_{i=2}^{r}\lambda_i\right)x_1\cdot\left(x_k-x_1\right)+\sum_{i=2}^{r}\lambda_i x_i\cdot\left(x_k-x_1\right)=$$

$$\lambda_1 x_1\cdot\left(x_k-x_1\right)+\sum_{i=2}^{r}\lambda_i x_i\cdot\left(x_k-x_1\right)=\sum_{i=1}^{r}\lambda_i x_i\cdot\left(x_k-x_1\right)=0.$$

Hay una ecuación por cada *k* desde 2 hasta *r*, disponiéndolas en forma matricial y agregando al principio la ecuación $\lambda_1 +\ldots+ \lambda_r = 1$ se concluye que el valor mínimo de *f* es una combinación de $\lambda$'s que satisface la siguiente ecuación:

$$\begin{pmatrix} 1 & 1 & \ldots & 1 \\ x_1\cdot\left(x_2-x_1\right) & x_2\cdot\left(x_2-x_1\right) & \ldots & x_r\cdot\left(x_2-x_1\right) \\ \vdots & \vdots & \vdots & \vdots \\ x_1\cdot\left(x_r-x_1\right) & x_2\cdot\left(x_r-x_1\right) & \ldots & x_r\cdot\left(x_r-x_1\right) \end{pmatrix}\begin{pmatrix}\lambda_1\\ \lambda_2\\ \vdots\\ \lambda_r\end{pmatrix}=\begin{pmatrix}1\\0\\0\\0\end{pmatrix} \quad (1.13)$$

o en forma abreviada:

$A_s\lambda = b$

Usando la regla de Cramer para resolver el sistema lineal se tiene que:

$\lambda_i = \Delta_i(Y_s)/\Delta(Y_s)$ en donde $\Delta(Y_s)$ es el determinante de $A_s$

De aquí que:

$$v(aff\ Y_s)=\sum_{i=1}^{r}\left(\Delta_i(Y)_s/\Delta(Y_s)\right)x_i.$$

*Lema* 1. $\Delta(Y_s)>0$ si y sólo si $Y_s$ es afínmente independiente, puesto que $\lambda_i$>0, $\Delta_i(Y)_s>0$ .

Para acceder a la demostración del *Lema* 1 ver referencia [3].

*Cálculo numérico*

Sea $Y_s=\{x_1,x_2\}=\{(-2,-2,0),(-1,-1,0)\}$

$$\begin{bmatrix} 1 & 1 \\ x_1\cdot\left(x_2-x_1\right) & x_2\cdot\left(x_2-x_1\right)\end{bmatrix}\begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix}=\begin{bmatrix}1\\0\end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ (-2,-2,0)\cdot(1,1,0) & (-1,-1,0)\cdot(1,1,0)\end{bmatrix}\begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix}=\begin{bmatrix}1\\0\end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ -4 & -2\end{bmatrix}\begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix}=\begin{bmatrix}1\\0\end{bmatrix}$$

$$\Delta(Y_s)=\begin{vmatrix} 1 & 1 \\ -4 & -2\end{vmatrix}=-2+4=2$$

$$\lambda_1=\frac{\Delta_1(Y_s)}{\Delta(Y_s)}=\frac{\begin{vmatrix}1&1\\0&-2\end{vmatrix}}{2}=\frac{-2}{2}$$

$$\lambda_2=\frac{\Delta_2(Y_s)}{\Delta(Y_s)}=\frac{\begin{vmatrix}1&1\\-4&0\end{vmatrix}}{2}=\frac{4}{2}$$

Como $\lambda_1$<0 no se puede construir $v(co\ Y_s)=\sum_{i=1}^{r}\lambda_i x_i,\quad x_i\in K,\quad \lambda_i>0,\quad \sum_{i=1}^{r}\lambda_i=1.$

*Subalgoritmo de la distancia*

Dado un conjunto finito, $Y_s=\{y_1,...,y_v\}\subset R^m$ y un ordenamiento de todos los subconjuntos de $Y_s$, *s* = 1, …,$\sigma$, se ejecutan los siguientes pasos:

1. Hacer *s* = 1;
2. si $\Delta(Y_s)>0$, $\Delta_i(Y)_s>0\ \forall i\in I$ y $\Delta_j\left(Y_s\cup\{y_j\}\right)\le 0\ \forall j\in I_s^{'}$, se define *v*(*co Y*) con ayuda de y (1.12) y parar.
3. Si *s* < $\sigma$, se incrementa *s* y se ejecuta el paso
4. En caso de que por errores de cálculo no se satisfagan las condiciones 1., 2., 3., detener la ejecución del algoritmo y devolver un mensaje de error.

Suponiendo que se obtiene con *v* = *m* + 1 y $Y_s$ = *Y*, entonces el convexo de *Y* es un simplex, cuando *v*(*co Y*) ∈ interior *co Y*. Luego *v*(*co Y*) = 0 y existe una esfera, contenida en *Y*, cuyo radio máximo con centro en el origen es $d^-$ dada por la distancia a la cara *m*−dimensional del *co Y* más próxima al origen, que representa una cota inferior de la distancia a la que debería trasladarse *co Y* si es que el origen debe permanecer fuera de *co Y*, así:

$$d^- = \min\left\{\left|v(aff\ Y_s)\right| : Y_s\subset Y \text{ tiene } m \text{ elementos}\right\} \quad (1.14)$$

$$\left|v(aff\ Y_s)\right|^2=\Delta(Y_s)^{-1}\sum\Delta_i(Y_s)y_i\cdot y_k,\ k\in I_s \quad (1.15)$$

*Procedimiento hacia atrás*

El algoritmo de la distancia no acumula los errores, pues en cada iteración *k*, $v_k = v(co\ V_k)$ es el resultado de la evaluación explícita de ecuaciones que dependen del conjunto $V_k$. Los errores provienen del cálculo de los productos internos, sobre todo cuando la distancia entre politopos es pequeña, del cálculo de $\lambda$ y del cálculo de $g_k$, este último puede reducirse si se sustituye el criterio de convergencia en el paso 3. del algoritmo que calcula la distancia por:

$$g_K(v_k)\le\varepsilon \quad (1.16)$$

Los autores proponen un algoritmo hacia atrás que permite calcular la distancia más corta entre 2 politopos cuando los errores de cálculo impiden concluir con éxito el subalgoritmo de la

distancia, el procedimiento, elige la mejor $Y_s$ y hace que $v(co\ Y) = v(aff\ Y_s)$:

$$v(co\ Y) = \arg\min \left\{ \begin{matrix} |y| : y = (aff\ Y_s), \\ s = 1,...,\sigma,\ \Delta(Y_s) > 0,\ j \in I_s \end{matrix} \right\} \quad (1.17)$$

## 4. Algoritmo numérico

Rich Rabbiz [8] codificó en C, el algoritmo GKJ que se ejecuta a continuación para el caso de dos cuadrados unitarios *A* y *B* con vértices en:

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad y \quad B = \begin{pmatrix} 2 & 2 & 0 \\ 3 & 2 & 0 \\ 3 & 3 & 0 \\ 2 & 3 & 0 \end{pmatrix}$$

1. Calcular un punto inicial $x \in K$ atendiendo a la diferencia de Mikownski:
   $P_0 = a_0 - b_0 = (-2 \quad -2 \quad 0)$
2. $-V_0 = -P_0 = (2 \quad 2 \quad 0)$
3. $G_P(-V_0) = (V_0 \square V_0) + h_A(-V_0) + h_B(V_0) = 4$
   $(V_0 \square V_0) = 8$
   $$h_A(-V_0) = \max \left\{ \begin{matrix} (0,0,0) \square (2,2,0) = 0 \\ (1,0,0) \square (2,2,0) = 2 \\ (1,1,0) \square (2,2,0) = 4 \\ (0,1,0) \square (2,2,0) = 2 \end{matrix} \right\} = 4,$$
   $s_A(-V_0) = (1,1,0)$
   $$h_B(V_0) = \max \left\{ \begin{matrix} (2,2,0) \square (-2,-2,0) = -8 \\ (3,2,0) \square (-2,-2,0) = -10 \\ (3,3,0) \square (-2,-2,0) = -12 \\ (2,3,0) \square (-2,-2,0) = -10 \end{matrix} \right\} = -8,$$
   $s_B(V_0) = (2,2,0)$
   $s_K(-V_0) = s_{K_A}(-V_0) - s_{K_B}(V_0) = (-1,-1,0)$
4. como $G_P > 0$ construye
   $V_1 = \{-V_0, (s_K(-V_0)\} =$
   $= \{(2,2,0), (s_A(-V_0) - s_B(V_0))\}$
   $V_1 = \{y_1, y_2\} = \{(2,2,0), (-1,-1,0)\}$
5. ejecuta el subalgoritmo para $V_1$:
   $Y_1 = \{y_1\}, Y_2 = \{y_2\}, Y_3 = \{y_1, y_2\}$
   $\Delta_1 Y_1 = \Delta_2 Y_2 = \Delta_3 Y_3 = 1$
   $\Delta_1 \{Y_2 \cup y_1\} = \Delta_2(Y_2) * (y_2 \square (y_2 - y_1)) =$
   $= 1 * ((-1,-1,0) \square (1,1,0)) = -2$
   $\Delta_2 \{Y_1 \cup y_2\} = \Delta_1(Y_1) * (y_1 \square (y_1 - y_2)) =$
   $= 1 * ((2,2,0) \square (1,1,0)) = 4$
   $\Delta(Y) = \Delta_1 \{Y_2 \cup y_1\} + \Delta_2 \{Y_1 \cup y_2\} = 2$
6. Dado que $\Delta_2 \{Y_1 \cup y_2\} > 0$, se construye
   $V_2 = -\lambda\, s_K(-V_0) + (0,0,0) =$
   $-\Delta_1 Y_1 / \Delta(Y) * (-1,-1,0) = (1,1,0)$
7. Como se trata de un solo punto, calcula $G_p$:
   $G_p = (V_2 \square V_2) + h_K(V_2) + h_K(-V_2)$
   $$G_p = 2 + \max \left\{ \begin{matrix} (0,0,0) \square (1,1,0) = 0 \\ (1,0,0) \square (1,1,0) = 1 \\ (1,1,0) \square (1,1,0) = 2 \\ (0,1,0) \square (1,1,0) = 1 \end{matrix} \right\} +$$
   $$+ \max \left\{ \begin{matrix} (2,2,0) \square (-1,-1,0) = -4 \\ (3,2,0) \square (-1,-1,0) = -5 \\ (3,3,0) \square (-1,-1,0) = -6 \\ (2,3,0) \square (-1,-1,0) = -5 \end{matrix} \right\} = 0$$
   $s_K(V_2) = s_A(V_2) - s_B(-V_2) =$
   $= (1,1,0) - (2,2,0) = (-1,-1,0)$
8. Se concluye que existe un plano de separación entre los politopos, la distancia mínima entre ellos es:
   $|v(co\ K)| = \sqrt{V_2 \square V_2} = \sqrt{2}$ entre los puntos $a = (1,1,0)$ y $b = (2,2,0)$

## 5. Conclusiones y trabajo futuro

La revisión cuidadosa del algoritmo presentado por GJK [3] permite comprender la razón de los cálculos que en él se realizan y como funcionan las variables de decisión:

$$G_p,\ \Delta(Y_s),\ \Delta_i(Y_s) \text{ y } \Delta_j \{Y_s \cup \{y_j\}\},\ j \in I_s^{'},$$

así mismo facilita la comprensión de este mismo algoritmo implementado en C por Rich Rabbiz [7] lo que facilitará su modificación y posterior incorporación al conjunto de subrutinas que permiten simular un objeto elástico virtual [4] que reacciona al contacto con la herramienta virtual que el usuario manipula mediante un dispositivo con retorno de fuerza o háptico.

## Agradecimientos

Este trabajo es parte del Proyecto HAPTICS realizado conjuntamente entre la UAS y el CINVESTAV con el apoyo del programa PROMEP. También se cuenta con apoyo del Consorcio RIBERO-PARTI del programa iberoamericano CYTED; en particular de los doctores R. Aracil de l DISAM-UPM y L. Basañez del IOC-UPC. Se agradecen los comentarios de W. Femín Guerrero Sánchez al presente texto.

## Referencias

[1] **Baraff**, D. "Curved Surfaces and Coherence for Non-penetrating Rigid Body Simulation". Computer Graphics. 24 [4] (1990) 19-28.

[2] **Edelsbrunner**, H. "Combinatorial Topology : Simplicial Complexes "*en*

www.cs.berkeley.edu /~jrs/meshpapers/edels (1999)

[3] **Gilbert**, E. G.; Johnson, D. W; Keerthi, S. S. "A Fast Procedure for Computing the Distance Between Complex Objects in Three−Dimensional Space". IEEE Journal of Robotics and Automation. 4 [2] (1988) 193 − 203.

[4] **Ibarra** Zannatha, J.M.; Marmolejo Rivas C. "Modelado de Cuerpos Virtuales 3D Elásticos para Mundos Virtuales con Retroalimentación Táctil" AmRob 2003.

[5] **Lin** M. C.; Manocha, D.; Ponamgi, M. "Fast Algorithms for penetration and contact determination between non-convex polyhedral models". IEEE Internacional Conference on Robotics and Automation 2707-2712 (1995)

[6] **Miritch** B. "V-CLIP - Collision detection algorithm for polyhedral objects" (1997) *en* http://www.cs.sunysb.edu/~algorith/implement/V-CLIP/implement.shtml

[7] **Moore** M.; Wilhelms, J. "Collision Detection and Response for Computer Animation". Computer Graphics. 22 [4] (1988), 289-298.

[8] **Rabbiz**, R. "Fast collision detection of moving convex poliyhedra" *en* http://www.cse.ucsc.edu/ ~pang /160/f98/Gems/GemsIV/collide.c

[9] **Rockafellar**, R. T.: Convex Analysis. Princeton University Press, Princeton New Jersey (1970) 451 pp.