SIMBOLIZZAZIONE AUTOMATICA DI OGGETTI AREALI E LINEARI

Claudio Rocchini (*)

(*) Istituto Geografico Militare, Via Cesare Battisti 10, 50122 Firenze
email: ad2prod@geomil.esercito.difesa.it

Riassunto
Questo lavoro presenta un compendio delle formule matematiche per il calcolo effettivo di lunghezze, aree, baricentri, direzioni principali (assi principali di inerzia), per linee ed aree, rappresentate da spezzate e poligoni (con eventuali buchi). Ogni formula è accompagnata da una breve introduzione teorica e da un'esauriente descrizione del suo effettivo utilizzo, soprattutto come strumento cartografico per la simbolizzazione automatica di feature areali e lineari. Le formule proposte sono state implementate come add-on dei software cartografici commerciali.

Abstract
This work introduces a mathematical compendium of the formulas for the effective calculation of lengths, areas, barycentres, directions (aces of inertia), for lines and areas, represented by polylines and polygons (with or without holes). Every formula is followed by a short theoretical explication and a description of its use as cartographic tool, for automatic symbolisation of areal-linear feature. The proposed formulas have been implemented as add-on of some commercial software.

Introduzione
Durante il processo di generalizzazione degli elementi cartografici, può accadere di dover sostituire un elemento areale o lineare con un elemento puntuale simbolizzato. E' il caso, ad esempio, della derivazione della carta 1:25,000 dell'I.G.M. a partire dalla cartografia tecnica regionale; infatti, mentre nella cartografia tecnica piscine e baracche sono elementi areali (e i ponti sono elementi lineari), nella cartografia I.G.M. questi oggetti devono essere rappresentati con simboli puntuali.
Questo lavoro presenta un metodo automatico per la riduzione di oggetti areali e lineari in oggetti puntuali con orientamento, i quali di solito sono associati ad un simbolo. La posizione e l'orientamento del simbolo sono calcolati in modo opportuno a partire dalle geometrie iniziali, in modo da rappresentare la posizione l'orientamento dell'oggetto originale.

Il posizionamento
Il primo passo per la trasformazione di una linea o di un'area in un punto orientato è costituito dalla scelta della posizione. La scelta naturale per la posizione di un punto è ovviamente quella del *baricentro* della linea o dell'area in questione. Vedremo nelle prossime sezioni come si possa calcolare in modo semplice il baricentro di qualsiasi poligono piano.

L'orientamento
Il secondo passo per la trasformazione in punto orientato è la scelta dell'orientamento del punto. La scelta che abbiamo fatto è quella di utilizzare come orientamento *l'asse principale di inerzia* della linea o dell'area in questione. L'asse principale di inerzia (o momento del secondo ordine) si dispone

lungo la direzione in cui è predominante la disposizione della massa dell'oggetto; quindi possiamo dire che descrive in modo corretto l'orientamento principale di una qualsiasi figura piana.
Altri strumenti software commerciali di generalizzazione scelgono invece di posizionare l'orientamento di una figura secondo l'asse della sua massima elongazione. In è possibile confrontare il risultato delle due soluzioni. Nelle prossime sezioni vedremo come sia possibile calcolare effettivamente la direzione principale di inerzia per ogni figura piana.

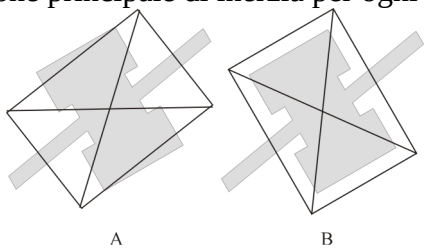

Figura 1. A) Disposizione lungo la direzione di massima elongazione (software commerciali). B) Disposizione secondo l'asse principale di inerzia (da noi utilizzata): la disposizione rispetta maggiormente l'andamento generale della figura.

Calcolo effettivo del baricentro e del momento: linee
Riportiamo adesso le formule effettive che permettono di calcolare lunghezza, baricentro e assi principali di inerzia di una qualsiasi polilinea. Descriviamo una polilinea *L* attraverso una lista di *n* vertici $P^1 \ldots P^n$.
Ricordiamo che con la scrittura ||*Q* - *P*|| si intende la distanza euclidea fra i punti *P* e *Q*, vale a dire:

$$||\bar{Q} - \bar{P}|| \stackrel{def}{=} \sqrt{(P_x - Q_x)^2 + (P_y - Q_y)^2} \tag{1}$$

Calcolo della lunghezza
La lunghezza di una polilinea *L* è data banalmente dalla somma delle lunghezze dei segmenti che la compongono, vale a dire:

$$L = \sum_{i=1}^{n-1} ||\bar{P}^{i+1} - \bar{P}^i|| \tag{2}$$

Calcolo del baricentro
Il baricentro si ottiene come media pesata dei baricentri dei singoli segmenti:

$$\bar{B} : \begin{cases} B_x = \frac{1}{2L} \sum_{i=1}^{n-1} ||\bar{P}^{i+1} - \bar{P}^i|| \cdot (P_x^{i+1} + P_x^i) \\ \\ B_y = \frac{1}{2L} \sum_{i=1}^{n-1} ||\bar{P}^{i+1} - \bar{P}^i|| \cdot (P_y^{i+1} + P_y^i) \end{cases} \tag{3}$$

dove *L* è la lunghezza totale della polilinea (formula precedente).

Calcolo del momento di inerzia
Il calcolo degli assi principali di inerzia è leggermente più complesso. Per prima cosa bisogna ricavare la matrice principale di inerzia che, in generale, è data da (omettendo le componenti *z*):

$$M = \int_S \begin{bmatrix} P_y^2 & -P_x P_y & 0 \\ -P_x P_y & P_x^2 & 0 \\ 0 & 0 & P_x^2 + P_y^2 \end{bmatrix} d\bar{P} \tag{4}$$

dove *S* è l'intera polilinea, e *P* un punto di essa. Gli elementi della matrice sono calcolati con le seguenti formule:

$$
\begin{aligned}
M11 &= \tfrac{1}{3A}\sum_{i=1}^{n-1}||\bar{P}^{i+1}-\bar{P}^{i}||\cdot\left[(P_x^i)^2+P_x^iP_x^{i+1}(P_x^{i+1})^2\right]\\
M12 &= \tfrac{1}{6A}\sum_{i=1}^{n-1}||\bar{P}^{i+1}-\bar{P}^{i}||\cdot\left[(-2P_y^{i+1}-P_y^i)P_x^{i+1}+(-P_y^{i+1}-2P_y^i)P_x^i\right]\\
M22 &= \tfrac{1}{3A}\sum_{i=1}^{n-1}||\bar{P}^{i+1}-\bar{P}^{i}||\cdot\left[(P_y^i)^2+P_y^iP_y^{i+1}(P_y^{i+1})^2\right]
\end{aligned}
\qquad (5)
$$

Dall'equazione (4) risulta evidente che $M_{12} = M_{21}$, inoltre non bisogna calcolare l'elemento M_{33}, dato che riguarda l'asse *z*. Una volta che si è ottenuto la matrice di inerzia possiamo passare a calcolare l'angolazione principale della polilinea:

$$
\begin{aligned}
\theta &= \arctan\tfrac{M_{12}}{M_{11}-M_{22}}\\
\rho_1 &= \tfrac{1}{2}|(M_{11}+M_{22})+(M_{11}-M_{22})\cos(\theta)+M_{12}\sin(\theta)|\\
\rho_2 &= \tfrac{1}{2}|(M_{11}+M_{22})+(M_{11}-M_{22})\cos(\theta+\Pi)+M_{12}\sin(\theta+\Pi)|\\
\theta_m &= \begin{cases}\rho_1 >= \rho_2 & \rightarrow \tfrac{1}{2}\theta\\ \rho_1 < \rho_2 & \rightarrow \tfrac{1}{2}\theta+\tfrac{\Pi}{2}\end{cases}
\end{aligned}
\qquad (6)
$$

Nell'equazione (6) il valore θ può rappresentare sia il doppio dell'angolazione dell'asse principale d'inerzia, sia il doppio dell'angolazione di quello secondario. Per distinguere i due casi è necessario calcolare la lunghezza ρ_1 dell'asse relativo, sia la lunghezza dell'asse perpendicolare ρ_2. A questo punto è possibile determinare l'angolazione principale d'inerzia θ_m. Per un approfondimento delle formule utilizzate si consulti un qualsiasi testo di Meccanica Superiore.

Calcolo effettivo del baricentro e del momento: aree

Riportiamo adesso le formule effettive che permetto di calcolare l'area, il baricentro e gli assi principali di inerzia di un qualsiasi poligono piano con buchi. Descriviamo un poligono *P* con buchi attraverso una lista di *m* curve poligonali semplici P^i di cui la prima rappresenta il bordo del poligono, mentre le altre rappresentano i bordi dei buchi interni (Figura 2).

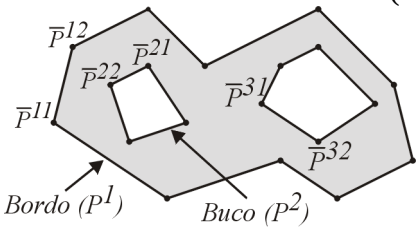

Figura 2. Rappresentazione di un poligono con buchi.

Ogni curva poligonale semplice P^i di *n* lati è quindi costituita da una serie di *n+1* vertici $\{P^{i1}, P^{i2}, \ldots, P^{in+1}\}$, in cui il primo vertice è uguale all'ultimo $P^{i1} = P^{in+1}$. Inoltre si adotta la convenzione di numerare i vertici del bordo esterno in *senso orario*; quelli dei buchi interni sono invece numerati in *senso antiorario*. In caso di errato orientamento dei vertici l'area del poligono (equazione (8)) risulterà negativa.

Ricordiamo infine che con il simbolo *X* indichiamo il prodotto vettoriale in R^2, vale a dire:

$$\bar{Q}\times\bar{W} \stackrel{def}{=} Q_xW_y - Q_yW_x \qquad (7)$$

Calcolo dell'area

L'area *A* di un poligono con buchi risulta essere uguale a:

$$A = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}(\bar{P}^{i\,j}\times\bar{P}^{i\,j+1}) \qquad (8)$$

Per una dimostrazione della correttezza di questa notevole formula si veda [2]. Le varie formule che seguono sono state dedotte dall'autore applicando il metodo di Franklin al calcolo del baricentro e della matrice di inerzia. Altri approfondimenti su questo argomento si possono trovare in [1] e [3]. E' importante sottolineare che la semplice equazione (8) funziona per qualsiasi poligono, concavo o convesso, e con qualsiasi disposizione di buchi.

Calcolo del baricentro
La posizione del baricentro del poligono è data dall'equazione:

$$\begin{aligned} B_x &= \tfrac{1}{6A}\sum_{i=1}^{m}\sum_{j=1}^{n}(\bar{P}^{i\,j}\times\bar{P}^{i\,j+1})(P_x^{i\,j}+P_x^{i\,j+1}) \\ B_y &= \tfrac{1}{6A}\sum_{i=1}^{m}\sum_{j=1}^{n}(\bar{P}^{i\,j}\times\bar{P}^{i\,j+1})(P_y^{i\,j}+P_y^{i\,j+1}) \end{aligned} \qquad (9)$$

dove *A* è l'area del poligono calcolata con (8).

Calcolo degli assi principali di inerzia
Come nel caso lineare, per prima cosa si calcolano gli elementi della matrice di inerzia:

$$\begin{aligned} K_{11}(i,j) &= P_x^{i\,j\,2}+P_x^{i\,j}P_x^{i\,j+1}+P_x^{i\,j+1\,2}-4B_x(P_x^{i\,j}+P_x^{i\,j+1})+6B_x^2 \\ M_{11} &= \tfrac{1}{12A}\sum_{i=1}^{m}\sum_{j=1}^{n}(\bar{P}^{i\,j}\times\bar{P}^{i\,j+1})K_{11}(i,j) \\ K_{22}(i,j) &= P_y^{i\,j\,2}+P_y^{i\,j}P_y^{i\,j+1}+P_y^{i\,j+1\,2}-4B_y(P_y^{i\,j}+P_y^{i\,j+1})+6B_y^2 \\ M_{22} &= \tfrac{1}{12A}\sum_{i=1}^{m}\sum_{j=1}^{n}(\bar{P}^{i\,j}\times\bar{P}^{i\,j+1})K_{22}(i,j) \\ K_{12}(i,j) &= (2P_x^{i\,j}-4B_x+P_x^{i\,j+1})P_y^{i\,j}+(P_x^{i\,j}-4B_x+2P_x^{i\,j+1})P_y^{i\,j+1} \\ &\quad -4(P_x^{i\,j}-3B_x+P_x^{i\,j+1})B_y \\ M_{12} &= \tfrac{1}{12A}\sum_{i=1}^{m}\sum_{j=1}^{n}(\bar{P}^{i\,j}\times\bar{P}^{i\,j+1})K_{12}(i,j) \end{aligned} \qquad (10)$$

A questo punto è possibile calcolare l'orientamento principale grazie all'equazione (6).

Gestione degli errori di calcolo
Dato che queste equazioni comprendono lunghe sommatorie di valori elevati a potenza e segni alterni, i calcoli risultano molto sensibili agli errori di calcolo dovuti alla limitata precisione di macchina. E' opportuno quindi normalizzare i dati a priori, scalando ad esempio i valori delle coordinate nell'intervallo *0-1* e quindi riportando i dati finali alla giusta scala. Questo accorgimento diviene indispensabile soprattutto se si lavora con dati in coordinate piane, che comportano valori numerici elevati, dovuti all'introduzione di un offset fisso (falsa origine). In fase d'implementazione è opportuno inoltre applicare algoritmi di calcolo sicuri, derivanti dalla teoria del calcolo numerico (es. somma stabile di numeri).

Cenni alla derivazione delle formule
Il calcolo dell'integrale di una formula su di un poligono, si può ricondurre alla somma degli integrali sui triangoli formati dai lati del poligono e dall'origine (Figura 3). Parametrizziamo un qualsiasi triangolo *0, v1, v2* con la seguente formula (sinistra) e relativo Jacobiano (centro), l'integrale di una qualsiasi funzione *f* sul triangolo risulta essere (destra):

$$\bar{P}(u,v)\mapsto \bar{v}^1\cdot u+\bar{v}^2\cdot v \quad J(\bar{P})=\begin{bmatrix} v_x^1 & v_x^2 \\ v_y^1 & v_y^2 \end{bmatrix} \quad \int_0^1\int_0^{1-u} f(\bar{P}(u,v))\cdot d(J(\bar{P}))dv\,du$$

Ad esempio per la coordinata x del baricentro si ha *f(P) = Px*, perciò si ha:

$$\int_0^1 \int_0^{1-u} \left(v_x^1\, u + v_x^2\, v\right)\left(v_x^1\, v_y^2 - v_x^2\, v_y^1\right) dv\, du =$$

$$= \int_0^1 \frac{1}{2}\, v_x^2\, \left(\bar{v}^1 \times \bar{v}^2\right)(1-u)^2 + v_x^1\, u\left(\bar{v}^1 \times \bar{v}^2\right)(1-u)\, du\; = \frac{1}{6}\, \left(\bar{v}^1 \times \bar{v}^2\right)\left(v_x^2 + v_x^1\right)$$

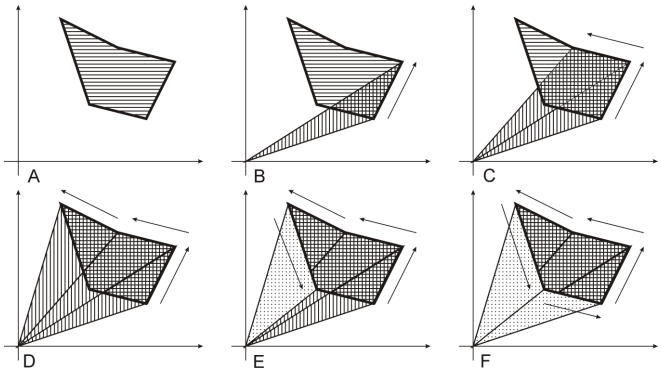

Figura 3.Calcolo dell'area di un poligono: si sommano le aree dei triangoli formati dall'unione di ogni lato del poligono con l'origine degli assi. I triangoli che "girano" nello stesso senso del poligono hanno un'area positiva (B,C,D), gli altri (E,F) negativa. La somma delle aree risulta uguale a quella del solo poligono originale (A).Il metodo funziona anche in presenza di buchi. Integrando una qualsiasi funzione sommabile su di un triangolo (baricentro, assi principali di inerzia, eccetera) si può applicare lo stesso procedimento per derivare la formula per l'intero poligono.

Risultati

Come esempio di realizzazione, si riporta il codice C++ per il calcolo delle formule (6), (9) e (10); il codice prevede la normalizzazione dei dati per evitare errori di calcolo:

```
struct point { double x,y; };
void orient( int np, point v[], double & area, point & baric, double & tetha ) {
    double minx,miny,maxx,maxy,scala;                  //Calcolo bbox per scalatura
    minx = maxx = v[0].x; miny = maxy = v[0].y;
    for(int i=1;i<np;++i) {
       if(minx>v[i].x) minx = v[i].x; if(miny>v[i].y) miny = v[i].y;
       if(maxx<v[i].x) maxx = v[i].x; if(maxy<v[i].y) maxy = v[i].y;
    }
    if( maxx-minx > maxy-miny ) scala = maxx - minx;
    else                        scala = maxy - miny;
    if(scala == 0) scala = 1;
    double bx = 0; double by = 0; double ar = 0;       //Calcolo area e baricentro
    for(i=0;i<np;++i){
        double p1x = (v[i  ].x - minx)/scala; double p1y = (v[i  ].y - miny)/scala;
        double p2x = (v[i+1].x - minx)/scala; double p2y = (v[i+1].y - miny)/scala;
        double cross = p1x * p2y - p2x * p1y;
        ar += cross; bx += cross*(p1x + p2x); by += cross*(p1y + p2y);
    }
    ar /= 2; bx /= ar * 6; by /= ar * 6;
    area  =   ar * scala * scala;                      // Dati Finali
    baric.x = bx * scala + minx; baric.y = by * scala + miny;
    double I11 = 0; double I12 = 0; double I22 = 0;    //Calcolo matrice momento
    double bx2 = bx*bx; double by2 = by*by;
    for(i=0;i<np;++i){
        double p1x = (v[i  ].x - minx)/scala; double p1y = (v[i  ].y - miny)/scala;
        double p2x = (v[i+1].x - minx)/scala; double p2y = (v[i+1].y - miny)/scala;
        double cross = p1x*p2y - p2x*p1y;
        I11 += cross * (p1x*p1x + p1x*p2x + p2x*p2x - 4*bx*(p1x+p2x) + 6*bx2);
        I22 += cross * (p1y*p1y + p1y*p2y + p2y*p2y - 4*by*(p1y+p2y) + 6*by2);
        I12 += cross * (p1y*(2*p1x-4*bx + p2x) + p2y*(p1x-4*bx+2*p2x) -4*by*(p1x-3*bx +p2x));
    }
    I11 /= ar*12; I22 /= ar*12; I12 /= ar*12;
    tetha = atan2(I12, I11 - I22) / 2;                 //Calcolo angolo prin. assi momento
    double elo1 = fabs((I11+I22+(I11-I22)*cos(2* tetha      )+I12*sin(2* tetha      ))/2);
    double elo2 = fabs((I11+I22+(I11-I22)*cos(2*(tetha+PI/2))+I12*sin(2*(tetha+PI/2)))/2);
    if(elo2 > elo1) tetha = tetha + PI / 2;
}
```

Per testare la metodologia proposta è stato implementato un applicativo software, in forma di add-on per prodotti commerciali, utilizzando il linguaggio C++.
L'utente deve scegliere la feature-class da simbolizzare (input) e la feature-class che conterrà il risultato della simbolizzazione (output). Altre opzioni permettono di copiare gli attributi del database o di non derivare l'orientamento (ad esempio, nella cartografia IGM DB25, i simboli di pozzo, aeroporto e campeggio non sono orientati). Nella Figura 1 sono riportati alcuni esempi reali di utilizzo, applicati alla trasformazione di cartografia regionale:

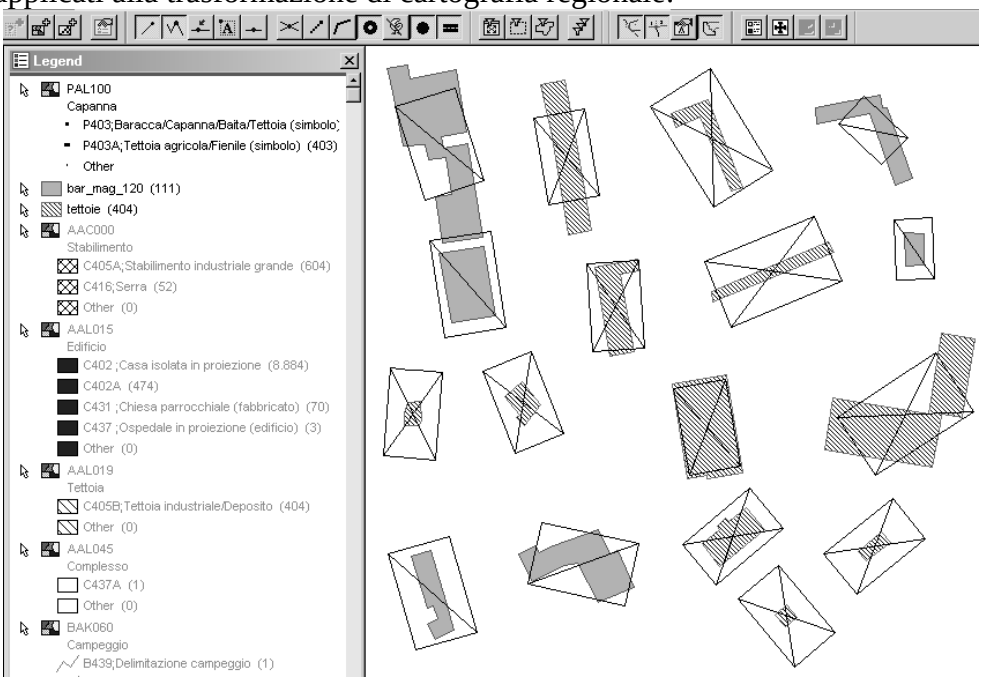

Figura 4. Esempio di applicazione della simbolizzazione (capanne, tettoie, etc.).

Per quanto riguarda le performance del software implementato, tutte le operazioni sono eseguite istantaneamente; le formule sopra descritte infatti, anche se a prima vista sembrano complesse, sono in realtà molto veloci da calcolare. Il software inoltre non alloca in alcun modo memoria aggiuntiva.

Conclusioni e Ringraziamenti

Abbiamo presentato un metodo automatico per la trasformazione di linee ed aree in simboli, che tiene in particolare conto la necessità di preservare l'orientamento generale della linea o dell'area da simbolizzare. Il metodo poi è stato testato grazie alla realizzazione di un applicativo software, realmente utilizzato nella catena produttiva dell'IGM.
Questo contributo è stato realizzato grazie alla fondamentale partecipazione di tutti i tecnici addetti ai servizi Fotogrammetrico, Cartografico ed Elaborazione Dati, dell'Istituto Geografico Militare, nonché utilizzando le attrezzature ed i dati dell'Istituto medesimo.

Bibliografia

[1] V. Akman, W.R. Franklin, M. Kankanhalli, and C. Narayanaswami. Geometric computing and uniform grid technique. *Computer-Aided Design*, 21(7):410–420, Sept. 1989.
[2] N. Chandrasekhar and W.R. Franklin. A fast practical parallel convex hull algorithm. Technical report, Electrical, Computer and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY, Nov. 1989.
[3] W.R. Franklin, N. Chandrasekhar, M. Kankanhalli, M. Seshan, and V. Akman. Efficiency of uniform grids for intersection detection on serial and parallel machines. In *New Trends in Computer Graphics – CGI '88*, pages 288–297, Geneva, Switzerland, 1988. Springer-Verlag.