INTRODUZIONE

Per animazione nel cinema si e' tradizionalmente intesa la tecnica di ripresa fotogramma per fotogramma di disegni e oggetti inanimati che permette di creare in proiezione l'illusione del movimento: si possono realizzare film d'animazione partendo da di segni, diagrammi, sagome articolate, pupazzi, modellini o addirittura disegnando i singoli fotogrammi direttamente sulla pellicola. L'animazione si realizza mediante la ripresa a passo uno di una serie di disegni leggermente diversi l'uno dall'altro in modo da rappresentare le diverse fasi del movimento.

Tale quadro e' vero anche oggi ma l'avvento di calcolatori sempre piu' potenti e della computer grafica ha rivoluzionato il modo di fare animazione. L'animazione 2D e' sempre stata affetta da numerose limitazioni che, seppure non precludendone l'utilizzazione, hanno sempre condizionato certune modalita' espressive. L'utilizzo del calcolatore ha permesso di superare molte di queste limitazioni, aumentando il potenziale tecnico e espressivo, modernizzando e rendendo piu' realistica la narrazione. Il grande successo di pubblico per produzioni cinematografiche con largo uso di queste tecniche moderne ha rafforzato l'utilizzo del computer nei lungome-

traggi come fornitore di effetti speciali e sequenze mozzafiato in una spirale di crescita vertiginosa che ha portato ad esempio la produzione Disney a toccare le vette piu' alte con "Toy Story", un lungometraggio interamente realizzato al computer.

Un problema centrale nella realizzazione di animazioni al calcolatore e' la fusione di forme: la metamorfosi di una forma in un'altra. Il problema consiste nel cercare di ridurre il numero di fotogrammi necessari per l'animazione considerandone un certo numero quali fondamentali e lasciando al calcolatore il compito di disegnare tutti quelli intermedi. Se si pensa che una buona animazione necessita di almeno 25 fotogrammi al secondo si capisce subito l'importanza e l'utilita' di un tale studio. Problema analogo e' la trasformazione (morphing) di una forma in una seconda non in relazione con essa. Si tratta di effetti ormai facilmente osservabili in film e produzioni video che tuttavia per essere prodotti necessitano ancora di un considerevole sforzo manuale.

Questa tesi propone diverse soluzioni ai problemi accennati sopra. Un interessante punto di partenza e' l'interpolazione lineare o mediante curve di Bezier. Essa pero' e' ben lungi dal fornire una soluzione accettabile. In seguito si presenta un algoritmo dovuto a Thomas W. Sederberg e altri e presentato nel 1992. Come vedremo l'algoritmo di Sederberg tuttavia soffre ancora di pesanti limitazioni. Infine si presenta un algoritmo di fusione di forme

dovuto a Shapira e Rappoport e presentato nel 1995. Esso rappresenta lo stato dell'arte riguardo al problema dell'interpolazione di forme piane al momento in cui si scrive. Tale algoritmo cerca di conservare forme e dimensioni durante l'animazione nonche' di seguire i percorsi che appaiono piu' logici nelle trasformazioni. Si tratta di metodi che lavorano su figure bidimensionali e che consentono di realizzare animazioni in tempo reale calcolatore permettendo.

Lo sviluppo tumultuoso dell'hardware dei calcolatori negli ultimi anni ha consentito di implementare tutti questi algoritmi su un sistema desktop con ottimi risultati. E' stato difatti utilizzato un sistema con processore AMD486 DX4 120 operante sotto il sistema operativo Windows '95 e con l'utilizzo del compilatore Borland C++ 4.51. Ognuno dei metodi presentati da' spunti per l'estensione al caso tridimensionale che non tardera' ad arrivare come sviluppo successivo.

Contenuti:

Capitolo 1: Si discute l'interpolazione lineare e la sua estensione con l'utilizzo di curve di Bezier;

Capitolo 2: Si discute l'algoritmo di animazione utilizzante un approccio fisico dovuto a Sederberg e altri;

Capitolo 3: Si discute l'algoritmo Star Skeleton dovuto a Shapira e Rappoport;

Capitolo 4: Descrizione dell'implementazione realizzata per questa tesi;

Appendice 1: Descrive un algoritmo per il calcolo del grafo di visibilita' di un poligono;

Appendice 2: Descrive un algoritmo per aggiungere punti di Steiner col metodo star skeleton.

CAPITOLO 1
INTERPOLAZIONE LINEARE
E CON CURVE DI BEZIER

1. INTERPOLAZIONE LINEARE

L'interpolazione e' un procedimento matematico mediante il quale si sostituisce a una distribuzione di valori osservati (o alla spezzata che la rappresenta graficamente) una distribuzione teorica (cioe' una curva continua detta curva interpolatrice) ottenuta in base a una funzione matematica. L'interpolazione ha quale scopo essenziale quello di colmare le lacune della distribuzione effettiva inserendo valori calcolati tra quelli ottenuti dall'osservazione. ∎

Il piu' semplice tipo di interpolazione e' quella lineare che consiste nel determinare una funzione di primo grado che assume due valori assegnati y_1 e y_2 in corrispondenza di due valori noti x_1 e x_2 della variabile indipendente. Con questa interpolazione si sostituiscono ai valori assunti dalla funzione $y = f(x)$ nell'intervallo (x_1, x_2) i valori corrispondenti che stanno sulla retta passante per (x_1, y_1) e per (x_2, y_2).

La formula

$$y = y1 + (x - x1) \cdot \frac{y_2 - y_1}{x_2 - x_1} \tag{1}$$

permette di ricavare y quando sia stato fissato il valore x.

Nel caso che stiamo trattando i valori osservati della distribuzione rappresentano le coordinate dei vertici di un poligono in un certo istante di tempo e in momenti successivi discreti. Si tratta quindi di costruire tutti i poligoni intermedi. La formula (1) fornisce la soluzione; bastera' infatti applicarla ripetutamente su tutti i vertici del poligono e al variare di x tra x_1 e x_2 si otterranno le coordinate dei vertici intermedi. Ricordando la rappresentazione parametrica di una retta la (1) si puo' riscrivere:

$$\begin{aligned} x &= (1 - t) \cdot x_2 + t \cdot x_1 \\ y &= (1 - t) \cdot y_2 + t \cdot y_1 \end{aligned} \tag{2}$$

dove stavolta facciamo variare t da 0 a 1.

Siano assegnati allora i poligoni $P_i = [P_{i0}, P_{i1}, \ldots, P_{in}]$ e $P_f = [P_{f0}, P_{f1}, \ldots, P_{fm}]$. P_i e' il poligono iniziale che dovra' essere trasformato in P_f mediante interpolazione lineare. Se $m \neq n$ sara' necessario far convergere piu' vertici della prima figura in un unico vertice della seconda o viceversa ma se $m = n$ potrebbe venire in mente di associare il vertice numero 0 della prima figura

al numero zero della seconda e cosi' via. Tale procedimento di corrispondenza automatica porta a risultati disastrosi specialmente se le figure sono state introdotte senza tenere in mente questa corrispondenza fissata per default. Poter effettuare trasformazioni decenti in modo automatico con la sola interpolazione tra vertici e' infatti una mera illusione. Nelle figure 1,2,3 si puo' vedere cosa accade interpolando in questo modo due poligoni introdotti dall'utente senza considerare la corrispondenza dei vertici che e' assegnata per default.

Il minimo indispensabile consiste allora nel fare impostare all'utente la corrispondenza tra i vertici dei due poligoni. Questo consentirebbe all'utente di scegliere tra l'animazione della figura 1 e quella della figura 2 scartando quella della figura 3. Se il numero dei vertici che costituiscono le figure inizia a diventare considerevole il tempo necessario per una corretta corrispondenza puo' diventare una considerevole parte di quello impiegato per lo sviluppo complessivo di un lavoro di animazione. In alcuni casi inoltre non esistera' una corrispondenza che non crei autointersezioni e sara' necessario introdurre manualmente alcuni fotogrammi intermedi.

Il calcolo dei fotogrammi intermedi puo' essere fatto come indicato dal programma 1.

```
for(float t=0; t¡=1; t+=passo)
{
  for(vertice=0; vertice<numv; vertice++)
  {
    vstart = corrispondenze[vertice][0];
    vend = corrispondenze[vertice][1];
    x1=coordinate[vstart][0];y1=coordinate[vstart][1];
    x2=coordinate[vend][0];y2=coordinate[vend][1]
    x = (1 − t)x2 + t · x1;
    y = (1 − t)y2 + t · y1;
    // qui si disegna il punto
  }
}
```

Programma 1: 'corrispondenze' contiene l'associazione tra i vertici della figura iniziale e finale. 'Passo' permette di variare il numero di fotogrammi.

2. INTERPOLAZIONE NON LINEARE

Una volta che la corrispondenza tra vertici e' data dall'utente o calcolata automaticamente rimane ancora da specificare il tipo di interpolazione. Abbiamo utilizzato quella lineare nel paragrafo

precedente ma questa e' solo una possibilita' poiche' si puo'scegliere una qualsiasi funzione come percorso.

2.1 LE CURVE IN COMPUTER GRAFICA

Una curva definita parametricamente in due dimensioni e' data da due funzioni di una variabile:

$$Q(u) = (X(u), Y(u))$$

dove $0 \leq u \leq 1$.

La rappresentazione parametrica e' molto piu' utile nella computer grafica rispetto la forma implicita la quale individua una curva in termini delle sue coordinate cartesiane, cioe':

$$f(x, y) = 0$$

Sia la rappresentazione parametrica che quella implicita sono rappresentazioni analitiche. In pratica questo significa che sono esatte e quindi le proprieta' della curva come la sua lunghezza sono estraibili da ciascuna descrizione. La rappresentazione parametrica e' pero' piu' utile per i nostri scopi perche' con essa fissato il parametro e' immediato ricavare le coordinate di un punto mentre in generale la forma implicita porta a risolvere una equazione

non lineare per ricavare le coordinate. Anche le derivate sono piu' naturalmente esprimibili in forma parametrica piuttosto che implicitamente. Le derivate espresse parametricamente descrivono i tassi di cambiamento rispetto alla direzione del parametro che cresce, cioe' lungo la curva, piuttosto che nelle direzioni della x crescente o della y crescente.

Una curva parametrica e' spesso in grafica un polinomio. Scriviamo un polinomio di ordine k+1:

$$Q(u) = p_0 + p_1 \cdot u + p_2 \cdot u^2 + \ldots + p_k \cdot u^k$$

Il grado del polinomio usato in computer grafica e' normalmente 3, cioe' e' cubico. Un polinomio di grado 2, quadratico, non offre suf- ficiente flessibilita' di forma. Per gradi maggiori di 3, c'e' un divario tra la descrizione, che e' molto piu' ingombrante, e la flessibilita' della forma. Curve di complessita' arbitraria possono essere descritte da polinomi di grado sufficientemente alto ma e' necessario un grande numero di coefficienti e possono essere introdotte delle oscillazioni non volute. Un polinomio di grado n puo' avere fino a n-1 punti di svolta se tutte le soluzioni di Q(u) sono reali. Questo implica che piu' grande e' n e piu' la curva oscilla. I polinomi con grado minore di 3 non possono essere fatti passare attraverso punti finali con specifiche proprieta' delle derivate. Questo e' un punto cruciale poiche' le curve composte,

fatte di segmenti di- stinti, necessitano di essere unite in modo "liscio" nei loro bordi. Quello cubico e' il polinomio di ordine piu' basso che fornisce continuita' C^1 e C^2. Per questi motivi ho scelto una classe di curve di grado tre per implementare una forma di interpolazione non lineare. Qualunque sia il grado del polinomio e' un inconveniente rappresentare la curva direttamente usando i coefficienti p_i. La relazione tra i coefficienti del polinomio e la forma della curva non e' chiara o intuitiva. Allora invece di manipolare i coefficienti direttamente, la forma polinomiale puo' essere determinata mediante punti di controllo e funzioni base che forniscono una piu' intuitiva connessione tra forma e curva.

LE CURVE DI BEZIER

Le curve di Bezier sono curve il cui polinomio cubico e' specificato indirettamente dai punti finali della curva e dai vettori tangenti a questi punti finali individuati da ulteriori due punti che non sono sulla curva. Siano P_1 e P_4 gli estremi della curva e P_2 e P_3 i punti esterni alla curva che individuano i vettori tangenti. Il polinomio cubico che rappresenta una curva di Bezier e' allora il seguente:

$$Q(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t)P_3 + t^3 P_4 \qquad (3)$$

INTERPOLAZIONE CON CURVE DI BEZIER

Le curve di Bezier come si e' visto sono facilmente rappresentabili: utilizzando l'equazione (3) e' molto semplice disegnarle. L'interpolazione attraverso curve di Bezier consiste nell'associare ad ogni coppia di vertici dei poligoni iniziale e finale una tale curva. In seguito per interpolare la figura si procedera' considerando la curva assegnata per ogni coppia di vertici. Questo si ottiene modificando il codice del programma 1 opportunamente. Si rimanda per questo al codice dell'applicazione associata alla presente tesi.

Utilizzare le curve di Bezier impone per l'utente lo sforzo ulteriore di introdurre una curva per ogni coppia di punti messi in corrispondenza tra il poligono di partenza e quello di arrivo. Sebbene questo possa fare perdere del tempo consente di estendere di molto le possibilita' di animazione rispetto all'interpolazione lineare che diventa solo un caso particolare. L'interfaccia grafica presentata con questa tesi consente all'utente di impostare col mouse i vettori tangenti che definiscono la curva di Bezier senza dover introdurre neanche un numero da tastiera. Per una descrizione piu' dettagliata rimando al capitolo 4. Le figure 4 e 5 indicano due animazioni realizzate mediante curve di Bezier.

CONCLUSIONE

Al di la' dell'intervento dell'utente per l'associazione dei vertici l'interpolazione soffre della pesante limitazione che se la seconda figura non e' sufficientemente vicina di forma rispetto la prima puo' non essere possibile ricreare l'animazione voluta comunque si assegnino le corrispondenze. L'interpolazione insomma puo' produrre risultati carenti anche per semplici forme e trasformazioni: i lati dei poligoni facilmente si intersecano tra loro causando una perdita del senso di fusione tra le forme. La principale ragione per la quale l'interpolazione tra vertici ottiene risultati poveri e' che ciascun cammino di vertice e' determinato indipendentemente da ogni altro e dipende solo dalle sue posizioni estreme.

CAPITOLO 2
UN APPROCCIO FISICO ALLA FUSIONE DI FORME IN 2D

1. IMPOSTAZIONE DEL PROBLEMA

Si consideri la situazione illustrata in figura 6. La fusione riportata praticamente giova molto poco eccetto che per illustrare come la fusione di forme puo' funzionare male con una semplice interpolazione con corrispondenze casuali tra vertici. In questo caso si ha un angolo che va a zero provocando la collisione tra due lati che si intersecano scavalcandosi. Un altro esempio di cattivo comportamento degli angoli intermedi e' dato dalla figura 7. Qui, gli angoli terminali dei vertici 4 e 6 sono entrambi 90° ma nei passi intermedi eccedono i 120°. Inoltre il vertice 5 inizia e finisce su una linea diritta che pero' si piega durante l'animazione. Le figure 6 e 7 suggeriscono due limitazioni sugli angoli che dovrebbero essere imposte sulle soluzioni di fusione. Primo, si dovrebbe evitare:

$$\theta_i(t) = 0 \qquad\qquad 0 \leq t \leq 1 \tag{1}$$

in ciascun vertice, poiche' questo implica che le figure intermedie si auto-intersecano. Secondo, e' preferibile, quando possibile, che

ciascun angolo intermedio sia limitato dai suoi angoli terminali. Cioe' θ_i(t) dovrebbe cambiare monotonicamente da $\theta_i(0)$ a $\theta_i(1)$.

E' possibile rappresentare il comportamento dell'angolo θ_i(t) in modo tale da poter dare una soluzione a queste condizioni. Siano P_i e P_j due punti. Ricordiamo che:

$$\mathbf{P_i} \times \mathbf{P_j} \equiv (\mathrm{x_i}, \mathrm{y_i}) \times (\mathrm{x_j}, \mathrm{y_j}) \equiv \mathrm{x_i y_j} - \mathrm{x_j y_i},$$

$$\mathbf{P_i} \cdot \mathbf{P_j} \equiv (\mathrm{x_i}, \mathrm{y_i}) \cdot (\mathrm{x_j}, \mathrm{y_j}) \equiv \mathrm{x_i x_j} + \mathrm{y_i y_j}.$$

e

$$\|\mathbf{P_i}\| = \sqrt{\mathrm{x_i^2} + \mathrm{y_i^2}}.$$

Ponendo u=1-t, l' angolo θ(t) puo' essere calcolato:

$$\begin{aligned} \theta_i(t) &= \angle[(\mathbf{P^0_{i-1}}\mathrm{u} + \mathbf{P^1_{i-1}}\mathrm{t}), (\mathbf{P^0_i}\mathrm{u} + \mathbf{P^1_i}\mathrm{t}), (\mathbf{P^0_{i+1}}\mathrm{u} + \mathbf{P^1_{i+1}}\mathrm{t})] \\ &= \angle[(\mathbf{B^0_i}\mathrm{u} + \mathbf{B^1_i}\mathrm{t}), 0, (\mathbf{F^0_i}\mathrm{u} + \mathbf{F^1_i}\mathrm{t})] \end{aligned} \tag{2}$$

dove $\mathbf{B^k_i} = \mathbf{P^k_{i-1}} - \mathbf{P^k_i}$ e $\mathbf{F^k_i} = \mathbf{P^k_{i+1}} - \mathbf{P^k_i}$ come mostrato in Figura 8 e dove P_i^0 e P_i^1 sono i vertici dei poligoni iniziale e finale. Ricordiamo che

$$\sin(\angle \mathbf{P_1}, 0, \mathbf{P_2}) = \frac{\mathbf{P_1} \times \mathbf{P_2}}{\|\mathbf{P_1}\| \, \|\mathbf{P_2}\|}$$

$$\cos(\angle \mathbf{P_1}, 0, \mathbf{P_2}) = \frac{\mathbf{P_1} \cdot \mathbf{P_2}}{\|\mathbf{P_1}\| \, \|\mathbf{P_2}\|}$$

$$\tan(\angle \mathbf{P_1}, 0, \mathbf{P_2}) = \frac{\mathbf{P_1} \times \mathbf{P_2}}{\mathbf{P_1} \cdot \mathbf{P_2}} \tag{3}$$

$$\begin{aligned} tan(\theta_i(t)) &= \frac{(\mathbf{F_i^0}(1-\mathrm{t}) + \mathbf{F_i^1 t}) \times (\mathbf{B_i^0}(1-\mathrm{t}) + \mathbf{B_i^1}\mathrm{t})}{(\mathbf{F_i^0}(1-\mathrm{t}) + \mathbf{F_i^1 t}) \cdot (\mathbf{B_i^0}(1-\mathrm{t}) + \mathbf{B_i^1}\mathrm{t})} \\ &= \frac{y_0(1-t)^2 + y_1 2t(1-5) + y_2 t^2}{x_0(1-t)^2 + x_1 2t(1-t) + x_2 t^2} \end{aligned} \tag{4}$$

dove:

$$x_0 = \mathbf{F_i^0} \cdot \mathbf{B_i^0}; \mathrm{x}_1 = \frac{\mathbf{F_i^1} \cdot \mathbf{B_i^0} + \mathbf{F_i^0} \cdot \mathbf{B_i^1}}{2}; \mathrm{x}_2 = \mathbf{F_i^1} \cdot \mathbf{B_i^1}; \tag{5}$$

$$x_0 = \mathbf{F_i^0} \times \mathbf{B_i^0}; \mathrm{x}_1 = \frac{\mathbf{F_i^1} \times \mathbf{B_i^0} + \mathbf{F_i^0} \times \mathbf{B_i^1}}{2}; \mathrm{x}_2 = \mathbf{F_i^1} \times \mathbf{B_i^1}; \tag{6}$$

L' equazione 4 puo' essere interpretata come una curva di Bezier di grado 2:

$$\begin{aligned} \mathbf{Q}(\mathrm{t}) &= (x_0, y_0)(1-t)^2 + (x_1, y_1)2t(1-t) + (x_2, y_2)t^2 \\ &= \mathbf{Q_0}(1-\mathrm{t})^2 + \mathbf{Q_1}2\mathrm{t}(1-\mathrm{t}) + \mathbf{Q_2}\mathrm{t}^2. \end{aligned} \tag{7}$$

Come illustrato in Figura 9, Q(t) ha l'importante proprieta' che $\theta_i(t) = \angle((1,0),(0,0),\mathbf{Q}(\mathrm{t}))$. Percio', $\theta_i(t) = 0$ solo se $\mathbf{Q}(\mathrm{t})$ interseca l'asse x positivo.

La monoticita' dell' angolo e' assicurata se nessuna linea che attraversa l'origine interseca $\mathbf{Q}(\mathrm{t})$ piu' di una volta (si veda la figura 16). La funzione angolo $\theta_i(t) = \angle[(1,0),(0,0),\mathbf{Q}(\mathrm{t})]$ ha

quattro estremi possibili: $\theta_i(0), \theta_i(1), \theta_i(t_1), \theta_i(t_2)$ dove t_1 e t_2 soddisfano l'equazione:

$$(\mathbf{Q}(\mathrm{t}) - \mathbf{0}) \times \mathbf{Q}'(\mathrm{t}) = 0.$$

Questo produce un polinomio cubico il cui grado si riduce sempre a un polinomio quadratico nella forma di Bernstein:

$$d(t) = d_0(1-t)^2 + d_1 2t(1-t) + d_2 t^2 = 0; \tag{8}$$

dove

$$d_0 = \mathbf{Q_0} \times \mathbf{Q_1};$$
$$d_1 = \frac{\mathbf{Q_0} \times \mathbf{Q_1}}{2};$$
$$d_2 = \mathbf{Q_1} \times \mathbf{Q_2}.$$

$\theta_i(\mathrm{t})$ cambia monoticamente se e solo se l'equazione 8 non ha radici reali nell' intevallo unitario.

L'algoritmo di Sederberg ci richiedera' di calcolare la variazione di angolo $\Delta\theta_i$. Se il triangolo $\Delta\mathbf{Q_0Q_1Q_2}$ non contine l'origine, allora $\Delta\theta_i = |\theta_i(1) - \theta_i(0)| mod 180^0$. Se il triangolo $\Delta\mathbf{Q_0Q_1Q_2}$ contiene l'origine, puo' accadere che $\Delta\theta_i$ eccede i 180^0 come mostrato in figura 10. Condizione necessaria e sufficiente affinche' $\Delta\theta_i$ ecceda i 180^0 e' che $\Delta\mathbf{Q_0Q_1Q_2}$ contenga l' origine, e $d_1^2 - d_0 d_2 < 0$ (il discriminate della formula quadratica nell' equazione 10).

Quindi:

$$\Delta\theta_i = \begin{cases} 360^0 - |\angle(\mathbf{Q_0}, (0,0), \mathbf{Q_2})| & se\ d_1^2 - d_0 d_2 < 0, \Delta\mathbf{Q_0 Q_1 Q_2} \supset (0,0) \\ |\angle(\mathbf{Q_0}, (\mathbf{0}, \mathbf{0}), \mathbf{Q_2})| & altrimenti \end{cases} \quad (9)$$

dove $\angle(\mathbf{Q_0}, (0,0), \mathbf{Q_2}) \leq 180^0$.

Se $\theta_i(t)$ non e' monotona, lo sviluppo nella sezione 3.2, necessita di conoscere *quanto lontano* $\theta_i(t)$ devia dalla monoticita'. Questa deviazione e' un angolo non negativo denotato da $\Delta\theta_i^*$ come mostrato in Figura 11.

2 VERTICI COINCIDENTI

I vertici coincidenti sono una occorrenza comune che necessita di una speciale attenzione. Quando n vertici adiacenti su un poligono si trovano sullo stesso punto, n-1 lati sull'altro poligono collassano a quel punto. Poiche' $\angle(\mathbf{P_{i-1}}, \mathbf{P_i}, \mathbf{P_{i+1}})$ e' indefinito se $\mathbf{P_i}$ e' coincidente con entrambi i suoi vicini, il cambiamento di angolo in un tale caso fornisce una fusione di forma anche indefinita. Vediamo come si puo' risolvere il problema.

Immaginiamo che i vertici coincidenti giacciono in uno spazio lungo la base di un triangolo isoscele infinitesimale, come mostrato in figura 13. In questo caso, $\theta_2 = \theta_4 = 90^0 + \frac{\alpha}{2}$ e $\theta_3 = 180^0$

in radianti. In generale, se i vertici $\mathbf{P_i}, \ldots, \mathbf{P_j}$ sono coincidenti, $\theta_i = \theta_y = 90^0 + \frac{\alpha}{2}$ e $\theta_{i+1} = \theta i + 2 = \ldots = \theta_{j-1} = 180^0$.

Nella figura 12, tutti i tre vertici di un poligono sono coincidenti. Comunque, come disegnato in figura 13, questi vertici sono trattati come infinitamente vicini ma separati lungo un segmento di linea. Percio', in tali casi, il punto di controllo $\mathbf{Q_2}$ o $\mathbf{Q_0}$ della curva $\mathbf{Q}$ sara' sempre posizionato in una distanza infinitesimale dall' origine lungo l'asse -x. La figura 14 mostra la curva $\mathbf{Q}$ per i vertici i=2 in Figura 12. In questo caso, $\mathbf{Q_2}$ giace su una distanza infinitesimale dall'origine lungo un raggio di 110^0 dall'asse +x come mostrato, e $\Delta\theta = 38^0$. La Figura 15 mostra un esempio di vertici coincidenti in cui un angolo intermedio va a zero.

3 MODELLI BASATI SULLA FISICA

La sezione 1 ha definito il problema della fusione di forme come essere quello di decidere dove aggiungere vertici nei due poligoni in maniera tale che poligoni intermedi nella fusione potessero essere definiti interpolando i vertici corrispondenti del problema assegna- to. La decisione su dove aggiungere i vertici deve essere guidata da qualche euristica. L' euristica che e' stata proposta e' quella di modellare un poligono $\mathbf{P^0}$ come un pezzo di filo fatto

di metallo idealizzato. La "migliore" fusione di forme e' quella che richiede il minimo lavoro per deformare $\mathbf{P^0}$ in $\mathbf{P^1}$ attraverso piegamenti e stiramenti.

Distinguiamo tra il lavoro che causa piegamento da quello che causa stiramento. Il lavoro di stiramento e' calcolato per ogni segmento di linea (cioe', ciascuna coppia di vertici adiacenti) mentre il lavoro di piegamento e' calcolato per ciascuna coppia di segmenti di linea adiacenti (cioe', per ciascun insieme di tre vertici adiacenti).

3.1 LAVORO DI STIRAMENTO

Una forza P stirera' un filo di lunghezza L_0 di un ammontare:

$$\delta = \frac{PL_0}{AE} \tag{10}$$

dove A e' l'area della sezione e E e' il modulo di elasticita', una costante del materiale (per esempio, E per l'acciaio e' 29,000,000 psi). Il lavoro speso per stirare un pezzo reale di filo di un ammontare δ e' :

$$W = \frac{\delta^2 AE}{2L_0} \tag{11}$$

Poiche' AE e' una costante del filo, per i nostri scopi la sostituiamo con una singola e definibile dall'utente "costante di durezza di sti-

ramento" k_s. Se L_0 e' la lunghezza iniziale della sezione di filo, e se L_1 e la sua lunghezza finale, l'equazione 11 calcolera' valori differenti se le forme iniziali e finali sono scambiate ($\frac{\delta^2 AE}{2L_0}$ in un caso e $\frac{\delta^2 AE}{2L_1}$ nell'altro). Inoltre, se un arco collassa ad un singolo vertice (cioe', L_0=0), l'equazione 11 richiede lavoro infinito. Queste due considerazioni motivano la seguente modifica all'equazione 11:

$$W_s = k_s \frac{(L_1 - L_0)^2}{(1 - c_s)min(L_0, L_1) + c_s max(L_0, L_1)} \tag{12}$$

dove $\delta = L_1 - L_0$ e c_s e' una costante definibile dall'utente che controlla la penalita' per gli archi che collassano in punti.

L'esponente 2 nelle equazioni (11) e (12) assume che il filo ha *elasticita' lineare*, che e' il caso se il filo non e' stato stirato troppo. Se si ha uno stiramento eccessivo, e' richiesto meno lavoro per allungare il filo perche' esso supera la *deformazione plastica* . In questo caso, un esponente piu' vicino a 1 esprime il lavoro speso. Percio', facciamo una modifica finale alla nostra equazione di stiramento:

$$W_s = k_s \frac{|L_1 - L_0|^{e_s}}{(1 - c_s)min(L_0, L_1) + c_s max(L_0, L_1)} \tag{13}$$

dove k_s, c_s, e_s sono costanti definibili dall'utente.

Nella realta' fisica, queste equazioni di lavoro hanno solo senso se il filo diventa piu' lungo ($L_1 > L_0$), non se diventa piu' corto

($L_1 < L_0$). Per i nostri scopi, calcoliamo sia il lavoro di stiramento che quello di compressione usando l'equazione (13).

Ponendo $L_0 = \|\mathbf{P_{i_1}} - \mathbf{P_{i_0}}\|$ e $L_1 = \|\mathbf{P_{j_1}} - \mathbf{P_{j_0}}\|$, noi denotiamo con:

$$W_s([i_0, j_0], [i_1, j_1]) = \frac{k_s |L_1 - L_0|^{e_s}}{(1 - c_s) min(L_0, L_1) + c_s max(L_0, L_1)} \tag{14}$$

il lavoro di stiramento richiesto per mappare $\mathbf{P_{i_0}} - \mathbf{P_{i_1}}$ in $\mathbf{P_{j_0}} - \mathbf{P_{j_1}}$, dove $i_1 = i_0$ o $i_1 = i_0 + 1$ e $j_1 = j_0$ o $j_1 = j_0 + 1$.

3.2 LAVORO DI PIEGAMENTO

Analogamente all'equazione per il lavoro di stiramento sviluppata nella sezione 3.1, il lavoro che causa piegamento e' definito nell'equazione 15 per il piegamento dell'angolo $\angle(\mathbf{P^0_{i_0}}, \mathbf{P^0_{i_1}}\mathbf{P^0_{i_2}})$ - nell' angolo $\angle(\mathbf{P^0_{j_0}}, \mathbf{P^0_{j_1}}\mathbf{P^0_{j_2}})$:

$$W_b([i_0, j_0], [i_1, j_1], [i_2, j_2]) = \tag{15}$$

$$\begin{cases} k_b(\Delta\theta + m_b\Delta\theta^*)^{e_b} & se\ \theta(t)\ non\ va\ mai\ a\ zero \\ k_b(\Delta\theta + m_b\Delta\theta^*)^{e_b} + p_b & se\ \theta(t)\ va\ a\ zero \end{cases}$$

dove $\Delta\theta$ e $\Delta\theta^*$ sono misurate in radianti e sono definite nel paragrafo 1. k_b, m_b, e_b, ep_b sono costanti definibili dall'utente. La costante k_b indica la durezza del piegamento, m_b penalizza gli

angoli che non sono monotoni, e_b e' un esponente che gioca un ruolo simile a e_s, e p_b penalizza gli angoli che vanno a zero.

3.3 NORMALIZZAZIONE

Osserviamo che il lavoro dovuto al piegamento e' indipendente dalla dimensione della forma. Percio', se le due forme sono scalate uniformemente, il calcolo del lavoro di piegamento non cambia. Comunque, il lavoro di stiramento varia con la scala della forma. Per rendere le costanti k_s e e_s indipendenti dalla scala, e' una buona idea mappare ciascuna forma in un rettangolo unitario, scalando dello stesso ammontare in x e y, cosi' che la dimensione piu' grande del lato del rettangolo sia uno. E' importante scalare uniformemente, o altrimenti gli angoli cambieranno, insieme con il calcolo del lavoro di piegamento.

E' interessante osservare che la scalatura uniforme della forma influisce sulla curva $\mathbf{Q}(\mathrm{t})$ ma non sul calcolo del lavoro di piegamento. Se $\mathbf{P^1}$ e' scalato di una costante c, allora $\mathbf{Q_0}$ non cambia, $\mathbf{Q_1}$ e' scalato da c, e $\mathbf{Q_2}$ e' scalata da c^2. Questo crea una differente curva $\mathbf{Q}(\mathrm{t})$, ma la funzione angolo $\angle((0,1),(0,0),\mathbf{Q}(\mathrm{t})) = \theta(\mathrm{t})$ non cambia.

4.0 IMPLEMENTAZIONE

Presentiamo finalmente un algoritmo che consente di implementare tutti i concetti introdotti nei paragrafi precedenti. Si tratta di un algoritmo semplificato che permette comunque ottimi risultati. Per l'algoritmo teorico completo rimando a [Sederberg].

Siano dati due poligoni P^0 e P^1 di m e n vertici rispettivamente. Tutte le corrispondenze tra vertici possono essere rappresentate in una matrice rettangolare mXn. Le colonne della matrice rappresentano i vertici di P^0 e le righe del grafo rappresentano i vertici su P^1. Il punto alla colonna i e riga j indica una corrispondenza tra P_i^0 e P_j^1.

Denotiamo con $[i, j]$ una corrispondenza tra P_i^0 e P_j^1 che puo' essere rappresentata sulla matrice come un punto nella giunzione della colonna i con la riga j. Una completa trasformazione di forma richiede che ad ogni vertice in P^0 corrisponda almeno un vertice in P^1 e viceversa. Inoltre permettiamo solo che $[i, j]$ e' una corrispondenza se $[i-1, j]$, $[i, j-1]$ o $[i-1, j-1]$ e' anche una corrispondenza. Poiche' $[0, 0] = [m, n]$ e' una corrispondenza, una soluzione completa puo' essere rappresentata nella matrice come un insieme di punti che vanno da $[0, 0]$ a $[m, n]$ con ciascun punto che si trova un passo ad est, sud o sudest rispetto al precedente. Non permettiamo che un passo a sud sia seguito immediatamente

da un passo a est, o che un passo ad est sia seguito da un passo a sud, poiche' tale combinazione e' piu' costosa di un singolo passo a sud est, eccetto sotto inusuali coefficienti di lavoro.

Se un punto $[i, j]$ giace su un cammino il precedente punto sul cammino e' indicato usando le funzioni ovest(i,j) e nord(i,j). Se il precedente punto e' direttamente a ovest di $[i, j]$ si ha ovest(i,j)=1 e nord(i,j)=0. Se il punto precedente e' sopra $[i, j]$ si ha ovest(i,j)=0 e nord(i,j)=1. Se il precedente punto e' a nord-ovest di $[i, j]$ si ha ovest(i,j)=1 e nord(i,j)=1.
Definiamo:

$$w_0 = W(i-1,j) + W_s([i-1,j],[i,j]) + \\ +W_b([i-1-ovest(i-1,j), j-nord(i-1,j)]), [i-1,j], [i,j])$$

$$w_1 = W(i,j-1) + W_s([i,j-1],[i,j]) + \\ +W_b([i-ovest(i,j-1), j-1-nord(i,j-1)], [i,j-1], [i,j])$$

$$w_2 = W(i-1,j-1) + W_s([i-1,j-1],[i,j]) + \\ +W_b([i-1-ovest(i-1,j-1), j-1-nord(i-1,j-1)], \\ [i-1,j-1],[i,j])$$

dove w_0 e' indefinita per i=0, w_1 e' indefinita per j=0, e w_2 e' indefinita per i=0 o j=0. Allora:

$$se\ w_0 \leq w_1, w_2: \quad W(i,j) = w_0; ovest(i,j) = 1; nord(i,j) = 0$$

$$se\ w_1 \leq w_0, w_2: \quad W(i,j) = w_1; ovest(i,j) = 0; nord(i,j) = 1$$

$se\ w_2 \leq w_0, w_1 : \quad W(i,j) = w_2; ovest(i,j) = 1; nord(i,j) = 1$

L'algoritmo per calcolare la soluzione a minimo lavoro consiste nel porre $W(0,0) = 0$ e calcolare W(i,j) dalle equazioni precedenti per $i = 0, \ldots, m; j = 0, \ldots, n$. W(m,n) e' allora la soluzione ottimale e le informazioni nord e ovest possono essere usate per tornare indietro e costruire il cammino che guida a questa soluzione ottimale.

CAPITOLO 3
ALGORITMO STAR SKELETON

1. INTRODUZIONE

E' stato visto che per definire una fusione di forme bisogna stabilire una corrispondenza tra vertici e una o piu' funzioni di interpolazione. Nel capitolo 1 si e' discusso di come dare la possibilita' all'utente di inserire tutti questi parametri. Naturalmente cio' comporta uno sforzo non indifferente da parte dell'utente. A cio' bisogna aggiungere che le animazioni non conservano normalmente le dimensioni delle figure da cui nascono come ad esempio le lunghezze dei lati o i valori degli angoli. L'algoritmo di Sederberg del capitolo 2 presenta un metodo di interpolazione potenziato perche' le entita' interpolate sono proprio le lunghezze dei lati e i valori degli angoli piuttosto che le sole posizioni dei vertici.

Questo metodo manipola molte situazioni con successo pero' in molti casi si producono autointersezioni dei contorni dei poligoni interpolati come si puo' vedere dalle figure 17 e 18. Inoltre il metodo tende a distorcere l'area del poligono nelle forme intermedie. Questo si ha perche' le differenti parti del contorno sono

distinte l'una dall'altra e l'area delle parti interne non e' esplicitamente considerata.

Per risolvere questi problemi il metodo star-skeleton suddivide in modo appropriato i poligoni da interpolare e rappresenta i sottopoligoni generati mediante coordinate polari. L'interpolazione viene effettuata tra queste coordinate polari. E' intuitivamente facile da comprendere il motivo di questa scelta. Le coordinate polari contengono esplicitamente piu' informazioni rispetto a quelle cartesiane rappresentando la figura mediante angoli e distanze.

2. DEFINIZIONI

Definizione 1. *Per poligono piano di n lati si intende la figura piana costituita da n punti* $P_1, P_2, \ldots, P_n$ *nell'ordine scritto e dai segmenti che li congiungono ordinatamente a due a due; i punti e i segmenti si dicono rispettivamente vertici e lati del poligono.*

Definizione 2. *Un poligono si dice convesso se e' tale che prolungando un qualsiasi lato il poligono resta tutto da una stessa parte. Altrimenti si dice concavo.*

Definizione 3. *Un poligono semplice e' un poligono i cui lati non si intersecano tra loro eccetto dove condividono un vertice.*

Definizione 4. *Un punto $v \in P$ si dice visibile da un punto $w \in P$ se il segmento di linea [v,w] e' contenuto in P (bordo incluso).*

Definizione 5. *Per poligono star si intende un poligono nel quale esiste un punto $v \in P$ dal quale sono visibili tutti gli altri punti di P. In un tale caso v si chiama punto star. Se un vertice e' un punto star questo e' chiamato vertice star.*

Definizione 6. *Un punto di Steiner e' un punto in un poligono che non e' un vertice.*

Definizione 7. *Una completa corrispondenza tra poligoni e' una funzione biunivoca che conserva l'adiacenza dei vertici e la loro orientazione.*

3. CHE COS'E' UNO STAR SKELETON

Assegnata una figura per scheletro intendiamo una struttura che consiste di una serie di collegamenti rigidi connessi da giunture che possono ruotare. Uno scheletro star e' uno scheletro costruito a partire da una decomposizione star di una figura insieme con i punti star stessi. L'insieme star e' un insieme di poligoni star chiamati pezzi star. Ciascun pezzo star possiede un punto star speciale chiamato origine star e una direzione chiamata direzione

di riferimento. Se c'e' piu' di un pezzo star nell'insieme star, ognuno possiede almeno un pezzo star come vicino, nel senso che essi condividono un lato. Tale lato e' chiamato lato condiviso, e i suoi vertici sono chiamati vertici condivisi. Un vertice condiviso deve essere interno al poligono P. I vertici condivisi non sempre sono vertici di P, poiche' i vertici dei pezzi star non sono necessariamente vertici di P. Due pezzi star possono intersecarsi tra loro.

I vertici di ciascun pezzo star sono rappresentati in coordinate polari rispetto la star origin del pezzo e la sua direzione di riferimento. Cioe', ciascun vertice e' definito dalla distanza dalla star origin e dall'angolo formato tra un vettore dalla star origin nella direzione di riferimento e il vettore dalla star origin al vertice.

Lo scheletro e' un albero planare composto di due tipi di punti: le star origin e i punti medi dei lati condivisi, chiamati punti medi. Ciascun vertice dello scheletro ha una direzione associata chiamata direzione di riferimento. La direzione di riferimento della radice e' l'asse x. La direzione di riferimento di ogni altro vertice e' il vettore dal vertice al suo genitore. La direzione di riferimento di un pezzo star e' la direzione di riferimento della sua origine star.

La radice e' rappresentata in coordinate cartesiane mentre ogni altro vertice e' rappresentato in coordinate polari rispetto al genitore e alla direzione di riferimento. Cioe', un vertice memorizza la distanza dal genitore e l'angolo tra la direzione di riferi-

mento del genitore e se stesso.

Gli angoli dei vertici dei pezzi star sono sempre misurati dalla direzione di riferimento del pezzo star. Comunque, rimane un ambiguita' a proposito dell'effettivo numero che rappresenta l'angolo, poiche' 2π puo' essere aggiunto o sottratto e l'angolo rimane lo stesso. La scelta del numero e' importante, poiche' due tali numeri sono interpolati per formare l'angolo in un poligono intermedio. Per evitare auto intersezioni del pezzo star in un poligono intermedio, gli angoli dei vertici star dovrebbero essre equivalentemente ordinati in entrambi i poligoni di input, e il loro range non dovrebbe eccedere 2π. Si ottiene questo denotando l'angolo di un vertice come la somma dell'angolo dal primo vertice della stella e l'angolo dal primo vertice alla direzione di riferimento (α). Cosi' gli angoli dei vertici sono equivalentemente ordinati in entrambi i poligoni e l'intervallo varia da α a $\alpha + 2\pi$.

3.1 COMPATIBILITA' TRA STAR SKELETON

Per interpolare gli scheletri associati al poligono iniziale e finale della fusione e necessario che gli scheletri siano compatibili. Un prerequisito per definire star skeleton compatibili e' una completa corrispondenza tra P e Q. Siano A e B due decomposizioni

star di due poligoni P e Q, rispettivamente. Denotiamo l'i-esimo pezzo star in A con A_i e lo j-esimo vertice di A_i con A_{ij} e analogamente per B. Assumiamo che esiste una corrispondenza biunivoca tra l'insieme V_a dei vertici di tutti i pezzi star in A e l'insieme V_b dei vertici di tutti i pezzi star in B. I vertici dei pezzi star che sono anche vertici di P o Q possiedono una corrispondenza dovuta alla completa corrispondenza richiesta. L'assunzione di corrispondenza indica che i vertici che non sono vertici di P o Q (vertici di Steiner) devono anche possedere tale corrispondenza. Diciamo che A e B sono compatibili se:

1. I vertici star di A e B sono composti di pezzi star compatibili.
   a) per ciascun j, A_{ij} corrisponde a B_{ij};
   b) le loro star origin sono combinazione convessa che coinvolge i vertici corrispondenti pesati con gli stessi pesi;
2. I due skeleton hanno esattamente la stessa struttura;

3.2 CALCOLO DI STAR SKELETON COMPATIBILI

Come primo passo si deve costruire una decomposizione star compatibile per i poligoni iniziale e finale. Le decomposizioni star di due poligoni sono compatibili se (1) ogni pezzo star in una decomposizione ha un compagno nell'altra decomposizione, che e'

definita da vertici corrispondenti;(2) ciascuna coppia di pezzi star corrispondenti ha almeno una coppia di vertici star corrispondenti. Il secondo criterio assicura che le star originin corrispondenti si trovano per ogni coppia di pezzi star.

Si noti che una coppia di poligoni P e Q potrebbe non avere una decomposizione star compatibile senza l'aggiunta di punti di Steiner. In un caso del genere si procede effettuando una decomposizione preliminare convessa del poligono iniziale P. Quindi si cerca un pezzo convesso A che ha un solo lato interno ab a P e si esamina il pezzo corrispondente B in Q. Se in quest'ultimo il lato ab risulta esterno allora si procede all'aggiunta dei punti di Steiner in Q e quindi alla decomposizione star del pezzo risultante. I punti di Steiner trovati devono essere poi associati in P con nuovi vertici. Conviene inserire questi ultimi sempre tra a e b. Nasceranno anche nuovi lati dalla decomposizione star che vanno riportati in P. Fatto cio' si considera un altro pezzo convesso in P con un solo lato interno non ancora esaminato e si ripete il processo. Se invece il pezzo associato in Q ad A ha il lato ab interno non e' necessario aggiungere punti di Steiner ma ci si limita ad effettuare la decomposizione star di B. La ragione per la quale si effettua una decomposizione convessa del poligono iniziale e' che le modifiche che nascono sul pezzo corrispondente nel poligono finale con l'aggiunta di eventuali punti di Steiner e per la decomposizione

star determinano una corrispondenza tra il sottopoligono finale e quello iniziale in ogni caso. Questo e' dovuto al fatto che in un pezzo convesso ogni vertice e' star point e quindi qualunque sia lo star point nel poligono finale il corrispondente nel poligono iniziale sara' star point. L'implementazione associata a questa tesi contiene due classi che consentono di effettuare rispettivamente una decomposizione star minimale e una decomposizione convessa di un generico poligono non intrecciato.

Nel momento in cui e' stata effettuata una decomposizione dei poligoni iniziale e finale in pezzi corrispondenti si determinano le star origin. Queste risultano essere il centro di massa degli star point corrispondenti per ogni sottopoligono. Questo assicura che la star origin e' uno star point.

Abbiamo ora tutto cio' che occorre per costruire lo scheletro. Questo e' costituito da star origin e punti medi. Nelle figura 19 sono rappresentate tre decomposizioni di poligoni con skeleton che non necessitano dell'aggiunta di punti di Steiner. Nelle figure 20,21 invece e' stato necessario aggiungere punti di Steiner. Lo scheletro unisce le star origin cosi' che il movimento relativo ai pezzi star durante la fusione delle forme puo' essere determinato. Aggiungiamo i punti medi per assicurare che lo scheletro giaccia completamente all'interno del poligono. Altrimenti, il suo moto potrebbe non riflettere il movimento della forma. Per costruire lo scheletro, si

devono vedere i pezzi star come costituenti un grafo planare, con un arco tra i nodi per i quali i pezzi star corrispondenti hanno un lato in comune. Un albero deve essere estratto da questo grafo planare. La cosa da fare qui e' scegliere una radice. La scelta della radice altera il movimento finale della forma. Si puo' lasciare all'utente la possibilita' di scegliere oppure automatizzare la scelta prendendo lo star point di un pezzo che ha il massimo numero di vicini. Durante la fusione, la posizione della radice e' semplicemente una interpolazione delle sue posizioni nel poligono iniziale e finale. Avendo lo scheletro si devono determinare le coordinate polari di ogni vertice delle figure. L'animazione verra' effettuata prima interpolando le coordinate polari e quindi calcolando le coordinate cartesiane associate che permetteranno di visualizzare l'animazione.

Nelle figure dalla 22 alla 26 sono visualizzati i risultati di alcune animazioni realizzate con l'applicativo associato a questa tesi.

3.3 IMPLEMENTAZIONE DI DECOMPOSIZIONI STAR

Descriveremo qui un algoritmo che consente di realizzare una

decomposizione star minimale di un poligono. Si tratta di un algoritmo che deve essere eseguito sui sottopoligoni determinati nel poligono finale. Nel caso in cui si voglia realizzare una decomposizione star compatibile dei poligoni iniziale e finale senza aggiungere punti di Steiner rimando a [Insalaco] e a [Shapira].

L'algoritmo usa la programmazione dinamica. Nella programmazione dinamica una soluzione ottimale e' trovata unendo soluzioni ottimali di sottoproblemi. Una soluzione ottimale a un sottoproblema e' rappresentata da uno stato con un costo associato. L'algoritmo procede in fasi che risolvono tutti i sottoproblemi di una certa dimensione usando le soluzioni calcolate nei passi precedenti.

Nel nostro caso, l'algoritmo costruisce una decomposizione minimale di un sottopoligono P da decomposizioni minimali dei sottopoligoni di P. I sottopoligoni sono definiti da vertici consecutivi $i, \ldots, j$ dove $j > i - 1$, e i vede j, e sono etichettati con P_{ij}. In ciascuna fase s l'algoritmo costruisce la decomposizione di sottopoligoni P_{ij}, in cui $j - i = s$. Ciascuno stato e' una decomposizione di un sottopoligono. Il costo dello stato e' la dimensione della decomposizione. In ciascuna decomposiziione M di un sottopoligono P_{ij} c'e un poligono star uno dei cui lati e' $[j, i]$. Chiamiamo questa stella la base sta della decomposizione M, che indichiamo con $S_{ij}(M)$. Il triangolo definito dai vertici i, m, j e'

denotato con T_{imj}. T_{imj} e' definito solo se $i < m < j$ e m vede sia i che j.

Uno stato e' costruito dagli stati delle fasi precedenti. La decomposizione M di P_{ij} e' costruita unendo una decomposizione A di P_{im} e una decomposizione B di P_{mj} per qualche m ($i < m < j$) che vede sia i che j (Figura 27). Questa unione puo' essere fatta in un numero di modi:

1. DoubleMerge - fonde T_{imj} con entrambe le basi star:

   $$M(P_{ij}) = (A(P_{im}) \cup S_{im}(A)) \cup (B(P_{mj}) \cup S_{mj}(B)) \cup \cup\{S_{im}(A) + S_{mj}(B) + T_{imj}\}$$

   $$size(M(P_{ij})) = size(A(P_{im})) + size(B(P_{mj})) - 1$$

2. SingleIMerge - fonde T_{imj} solo con $S_{im}(A)$:

   $$M(P_{ij}) = (A(P_{im}) \cup S_{im}(A)) \cup B(P_{mj}) \cup \{S_{im}(A) + T_{imj}\}$$

   $$size(M(P_{ij})) = size(A(P_{im})) + size(B(P_{mj}))$$

3. SingleJMerge - fonde T_{imj} solo con $S_{mj}(B)$:

   $$M(P_{ij}) = A(P_{im}) \cup (B(P_{mj} \cup S_{mj}(B)) \cup \{S_{mj}(B) + T_{imj}\}$$

   $$size(M(P_{ij}) = size(A(P_{im})) + size(B(P_{mj}))$$

4. NoMerge - non fonde T_{imj} con nessuna base star

   $$M(P_{ij}) = A(P_{im}) \cup B(P_{mj}) \cup \{T_{imj}\}$$

   $$size(M(P_{ij}) = size(A(P_{im}) + size(B(P_{mj})) + 1$$

   Per ciascun sottopoligono P_{ij} esistono due tipi di stati:

   $M(P_{ij})$: Una decomposizione star di P_{ij}. C'e un solo tale stato per ciascun sottopoligono. Questa e' una decomposizione

minima tra tutte le decomposizioni star di P_{ij}

$M_x(P_{ij})$: Una decomposizione minimale di P_{ij} da x. C'e un solo tale stato per ciascuna combinazione di un sottopoligono e un vertice. C'e una decomposiziione minimale di P_{ij} tra tutte le decomposizioni la cui base star e' vista da x. La base star potrebbe non essere una stella, cioe' x potrebbe non essere nella base star. Gli altri elementi nella decomposizione eccetto la base star sono tutte star.

Due decomposizioni A e B di un sottopoligono P_{im}, che hanno differenti basi star, hanno differenti capacita' di merge. Potrebbe accadere, per esempio, che $S_{im}(A) + T_{imj}$ e' una star, mentre $S_{im}(B) + T_{imj}$ non lo e'. In questo caso, $S_{im}(A)$ puo' essere fuso con SingleIMerge con T_{imj}, mentre $S_{im}(B)$ non puo' esserlo. Percio', non e' sufficiente conservare una decomposizione minimale per ciascun sottopoligono. Per ciascuna possibile base star S, dobbiamo tenere una decomposizione minimale tra tutte le decomposizioni la cui base star e' S. Poiche' noi siamo interessati solo nelle stelle aventi un vertice che e' un punto star, e' sufficiente tenere per ciascuna coppia (P_{ij}, x), una decomposizione minimale di P_{ij} la cui base star e' vista da x.

3.4 DECOMPOSIZIONE CONVESSA

Realizzare un algoritmo per decomporre in pezzi convessi un poligono e' possibile se si considera il legame tra la decomposizione convessa e la decomposizione star. La decomposizione convessa e' una decomposizione star in cui si aggiunge il vincolo che in un pezzo risultante ogni vertice dev'essere star. Cioe' un pezzo convesso e' un pezzo in cui ogni vertice e' uno star point. Da questo si intuisce che l'algoritmo di decomposizione convessa sara' basato sulla stessa idea della decomposizione star: si tratta di un algoritmo di programmazione dinamica che procede allo stesso modo imponendo pero' la restrizione che ogni vertice dev'essere star per poter effettuare una fusione.

CAPITOLO 4
DESCRIZIONE DELL'IMPLEMENTAZIONE

Nel momento in cui si avvia il programma si puo' decidere se realizzare un nuovo progetto o caricarne uno vecchio da disco. Nel primo caso dal menu **File** si deve scegliere **New** altrimenti sempre dal menu **File** bisogna scegliere **Apri**. Quando si carica un progetto gia' realizzato si considera come tipo di animazione quella per cui il progetto e' stato pensato: se ho creato un progetto per una animazione di tipo Star e lo memorizzo allora quando verra' ricaricato ci ritroveremo automaticamente in modalita' di animazione Star.

Se si crea un nuovo progetto viene presentato un dialog che consente di scegliere il tipo di animazione su cui si basera' quel progetto. Le alternative sono tre: 1. Interpolazione, 2. Sederberg, 3. Star skeleton. Dopo aver fatto questa scelta si entra nell'editor per introdurre i poligoni.

1.1 L'EDITOR

Nella sua configurazione iniziale l'editor presenta due finestre

che coprono le due meta' dell'area di lavoro. E' possibile spostare a piacimento o ridimensionare le due finestre ed e' anche possibile aprire nuovi progetti ritrovandosi a lavorare con 4,6 o piu' finestre sullo schermo. Questo e' possibile perche' l'applicazione realizzata e' MDI (Multiple Document Interface) il che consente il vantaggio di poter lavorare su piu' progetti contemporaneamente. Ogni finestra indica il tipo di algoritmo ad essa associato e un numero d'ordine che si rivela assai utile se sono aperti piu' progetti con lo stesso algoritmo associato.

Supponiamo di avere due sole finestre con associato uno dei tre metodi di animazione possibile. Per introdurre un poligono si ci deve posizionare su una delle due finestre ed inserire i punti premendo il tasto sinistro del mouse. Non vi e' distinzione tra finestra associata al poligono iniziale e finestra associata al poligono finale. Infatti il poligono iniziale sara' quello associato alla finestra attiva nel momento in cui si fa partire l'animazione.

Le opzioni di editor sono fondamentalmente 3:

(1) Inserimento di nuovi vertici;

(2) eliminazione di vertici;

(3) Spostamento di vertici.

1.1.1 Inserimento di nuovi vertici

L'editor e' espressamente progettato per introdurre poligoni semplici. Questo vuol dire che a mano a mano che si inseriscono i punti questi vengono automaticamente collegati coi punti immediatamente precedenti se non si generano intersezioni. Per questa ragione se introduciamo due vertici a e b non sara' piu' possibile inserire vertici tra a e b col mouse perche' nascerebbe un poligono non semplice. Se si vogliono inserire nuovi vertici si deve invece cliccare sul pulsante che presenta l'icona per la introduzione di nuovi vertici: un dialog allora consentira' di introdurre dove si vuole un nuovo vertice.

1.1.2 Eliminazione dei vertici

Per eliminare uno o piu' vertici introdotti si deve premere e tenere premuto il pulsante destro del mouse. Trascinando il mouse apparira' un rettangolo di selezione. Per eliminare dei vertici e' necessario che questi ricadano dentro questo rettangolo. Dopo di cio' si rilascia il pulsante destro e apparira' accanto al cursore un menu volante: scegliere **elimina**. I vertici selezionati allora spariranno.

1.1.3 Spostamento di vertici

E' possibile spostare solo un vertice alla volta tra quelli gia' presenti in figura. Per selezionarlo si procede come nel caso dell'eliminazione dei vertici. In questo caso pero' dal menu volante scegliere **sposta**. Il vertice selezionato sara' trascinabile col solo movimento del mouse. Quando si e' raggiunta la posizione desiderata premere il tasto destro del mouse per imporre quella posizione come definitiva.

1.2 L'ANIMAZIONE

Per realizzare l'animazione ci si deve posizionare sulla finestra associata al poligono di inizio. Quindi si deve cliccare il pulsante destro del mouse e scegliere dal menu volante l'opzione **anima**. A questo punto apparira' l'animazione desiderata se il progetto e' di tipo Star Skeleton. Se il progetto e' di interpolazione invece si dovranno inserire le corrispondenze tra i vertici ed eventualmente le curve di Bezier.

Se il progetto e' di tipo Sederberg si dovranno introdurre i parametri numerici relativi ad esso.

1.2.1 LE CURVE DI BEZIER

Se si desidera realizzare una animazione con interpolazione ma scegliendo curve di Bezier come funzioni interpolatrici prima di scegliere anima dal menu volante si deve individuare il menu con la scritta **bezier** e selezionare l'unica voce presente: questo consentira' di attivare la modalita' di animazione con curve di Bezier. Per ritornare alla modalita' normale basta effettuare la stessa operazione.

A questo punto dal menu volante scegliere **anima**: oltre che alla introduzione delle corrispondenze apparira' una nuova finestra che permette di associare una curva di Bezier ad ogni coppia di vertici dei poligoni. Con il pulsante sinistro del mouse si sceglie su quale vettore associato agli estremi della curva intervenire mentre con il pulsante destro del mouse si va avanti e si ha la possibilita' di introdurre le altre curve. Una volta finita questa fase partira' l'animazione.

APPENDICE 1
COME COSTRUIRE IL GRAFO DI VISIBILITA' DI UN POLIGONO

INTRODUZIONE

Sia assegnato un poligono semplice $P = [P_0, P_1, \ldots, P_n]$. Il problema della costruzione del grafo di visibilita' e' fondamentale per eseguire una qualsiasi decomposizione di un poligono in sottopoligoni. Nel contesto di questa tesi risulta fondamentale per determinare la decomposizione convessa o la decomposizione star di un poligono. In questa appendice verra' spiegato il metodo che ho sviluppato per ottenere il risultato voluto.

SVILUPPO

Sia assegnato un vertice v_1 del poligono P. Siano v_0 e v_2 i vertici adiacenti a v_1. Chiameremo area di visibilita' del vertice v_1 quella porzione di piano delimitata dalle due semirette nascenti da v_1 e passanti per v_0 e v_2 e tale che la parte interna del poligono piu'

vicina al vertice v_1 faccia parte di essa . In Figura 29 e' indicata la costruzione per stabilire come scegliere il semipiano corretto tra i due possibili.

Tutti i vertici del poligono che ricadono nell'area di visibilita' potrebbero essere visti o meno dal vertice v_1. Sia infatti v_i ricadente nell'area di visibilita' di v_1. Se si trova un lato del poligono tra loro v_i non sara' visibile da v_1 altrimenti sara' visibile. Se un vertice non ricade nell'area di visibilita' di v_1 allora siamo sicuri che non risultera' visibile ed abbiamo un risultato definitivo da inserire nella matrice di visibilita'. A tal proposito si veda la figura 29. Questo test va applicato su tutti i vertici del poligono. Si noti che se il vertice v_a cade nell'area di visibilita' del vertice v_b non e' detto che valga il contrario e in quest'ultimo caso siamo sicuri che v_a non vede v_b.

Dopo questa prima fase la matrice di visibilita' contiene dei valori certi associati a vertici che sicuramente non si vedono tra loro: ho posto questi elementi pari a -1. Per gli altri vertici invece non si puo' stabilire ancora nulla. Ho posto le posizioni corrispondenti uguali a 2.

La prima fase consente di riempire molte posizioni della matrice di visibilita' soprattutto se il poligono cresce di complessita'. Una volta determinata l'area di visibilita' e' facile determinare il semipiano su cui giace un generico vertice e quindi se e' non visi-

bile: basta controllare da che parte delle semirette di trova.

Nella seconda fase vengono presi in considerazione tutte le coppie di vertici associate a posizioni della matrice con valori uguali a 2. Siano v_a e v_b due tali vertici. Se tra loro si trova un lato del poligono distinto da quelli a loro adiacenti, cioe' se il segmento s che ha per estremi v_a e v_b interseca un lato del poligono in un punto che non sia un suo estremo allora i due vertici non si vedono e si va avanti. Se il segmento invece non interseca nessun lato del poligono i due vertici si vedono. Resta da considerare tutta una serie di casi particolari mostrati in figura 31. Se il segmento s interseca un vertice, per esempio, allora si potrebbe uscire dal poligono passando per questo vertice oppure restare ancora all'interno; se un lato del poligono giace sul segmento s si potrebbe anche uscire o entrare dal poligono.

Supponiamo che il lato l del poligono generato dai vertici v_i e v_f giaccia sul segmento s. Siano v_{ii} e v_{ff} i vertici precedente e successivo rispettivamente di v_i e v_f. (1) Se questi vertici si trovano su lati opposti di s allora i vertici v_a e v_b non si vedono; (2) Se v_{ii} giace anche su s ma v_{ff} non si controlla qual'e' il campo di visibilita' di v_f. Se questo e' il semipiano piu' piccolo allora i vertici non si vedono . (3) Se v_{ii} e v_{ff} giacciono dalla stessa parte di s allora il lato l non li nasconde in virtu' del fatto che v_a cade nel campo di visibilita' di v_b e viceversa.

Supponiamo che il lato l del poligono generato dai vertici v_i e v_f abbia il vertice v_i in comune con s. Si prende in esame v_{ii}. Se v_{ii} e v_f sono su lati opposti di s v_a e v_b non si vedono. Se v_{ii} giace anch'esso sulla retta s allora se il campo di visibilita' di v_i e' il semipiano piu' piccolo non si vedono. Sempre per il fatto che v_a cade nel campo di visibilita' di v_b e viceversa allora l non nasconde v_b se il campo di visibilita' di v_i e' il semipiano piu' grande tra i due possibili. Si veda la figura 30

Concludo osservando che le informazioni raccolte nel primo passo dell'algoritmo sono essenziali per il corretto funzionamento della seconda parte.

APPENDICE 2
COME AGGIUNGERE PUNTI DI STEINER

In questa breve appendice verra' descritto come e' possibile trovare i punti di Steiner nel poligono finale nel processo di decomposizione necessario nell'algoritmo Star Skeleton.

METODO PRATICO

Come e' noto, nell'algoritmo star skeleton, dopo aver effettuato una decomposizione convessa del poligono iniziale si devono studiare i sottopoligoni che nascono nel poligono finale in corrispondenza di tali pezzi convessi. Se un pezzo convesso corrisponde a un sottopoligono interno al poligono finale non e' necessario aggiungere punti di Steiner e si procede con una decomposizione star del pezzo ma nel caso in cui il sottopoligono non e' lecito sara' necessario aggiungere punti di Steiner: vediamo come.

Sia A il pezzo convesso scelto dal poligono iniziale P e sia ab il suo lato interno. Sia B il sottopoligono corrispondente nel poligono finale Q. Puo' accadere che il lato ab sia esterno a B in Q . In questo caso vanno aggiunti punti di Steiner. Per introdurre i punti di Steiner si dovra' seguire un percorso in B che parte da

a e raggiunge b attraverso i restanti vertici $v_1, \ldots v_{n-1}$ di B. Si associa un punto di Steiner inizialmente ad ognuno di tali vertici incontrati nell'ordine e li si congiunge con segmenti. Poiche' stiamo andando da a a b nascera' un sottopoligono di Q che associeremo ad A. Il problema qui sta nello scegliere la parte giusta dalla quale muoversi per aggiungere i punti di Steiner: questi infatti devono essere interni a Q. E' necessario avere allora informazioni su da che parte dei vertici a e b si e' all'interno di B. Queste informazioni si possono recuperare da quelle ricavate per costruire il grafo di visibilita' del poligono dove erano indispensabili per costruire la matrice di visibilita'. In figura 31 si vede come si procede.

Una volta individuati tutti i punti di Steiner se ne dovra' eliminare uno per ogni tre consecutivi che si vedono. Se s_1, s_2, s_3 si vedono tra loro allora s_2 andra' eliminato perche' superfluo. Questo rende necessario la costruzione di una matrice di visibilita' per il sottopoligono di Q nato da questi punti di Steiner: si puo' richiamare allora il codice gia' scritto per la costruzione del grafo di visibilita' di un generico poligono semplice.