UNIVERSITA' DI PISA

Facoltà di Ingegneria

Corso di Laurea Specialistica in Ingegneria Informatica
per la Gestione d'Azienda

TESI DI LAUREA

Ricostruzione Di Oggetti Tridimensionali Basata su Sistemi Fuzzy a Partire da Punti Acquisiti Mediante Scanner

RELATORI

Prof. Beatrice LAZZERINI

Ing. Marco COCOCCIONI

CANDIDATO

Luisa COLUCCIA

Anno accademico 2012/2013

"Ognuno è un genio.

Ma se si giudica un pesce

dalla sua abilità di arrampicarsi sugli alberi

lui passerà tutta la sua vita a credersi stupido."

Albert Einstein

INDICE

SOMMARIO **5**

1 INTRODUZIONE **6**

1.1 SCANSIONE DI OGGETTI 3D 9

1.1.1 Digitalizzazione *11*

1.1.2 Generalizzazione *12*

1.1.3 Integrazione di scansioni multiple *13*

1.1.4 Ottimizzazione *14*

2 RICOSTRUZIONE DI SUPERFICI **15**

2.1 COSTRUZIONE DELLA SUPERFICIE DA UNA NUVOLA DI PUNTI 15

2.2 TECNICHE DI SUDDIVISIONE DELLO SPAZIO 16

2.3 METODI DI SURFACE FITTING 18

3 IL SURFACE FITTING COME PROBLEMA DI REGRESSIONE **22**

3.1 PROBLEMI DI REGRESSIONE 22

3.2 APPROCCI PARAMETRICI VS APPROCCI NON PARAMETRICI 23

3.3 REGRESSIONE BASATA SU ISTANZE (INSTANCE-BASED REGRESSION) 25

3.4 REGRESSIONE LOCALMENTE PONDERATA (LOCALLY WEIGHTED REGRESSION) 26

3.5 RETI NEURALI ARTIFICIALI 27

3.6 REGRESSIONE TRAMITE L'INDUZIONE SU REGOLE (RULE INDUCTION REGRESSION) 30

4 SISTEMI A REGOLE FUZZY **33**

4.1 IL PROCESSO DI INFERENZA FUZZY 35

4.2 APPROCCIO TAKAGI-SUGENO 36

5 UN METODO VELOCE ED EFFICIENTE PER L'INFERENZA FUZZY **38**

5.1 PASSI DELL'ALGORITMO 38

5.2 FUNZIONE FITS2D O FITS3D 39

5.3 FUNZIONE "EVALFIS_SIRBT1TS0_F" 40

5.4 MARCHING CUBES E MARCHING SQUARES 41

6 ESPERIMENTI IN $\Re^2$ **43**

6.1 PCREC2 43

6.1.1 Creazione del dataset: funzione circR1_withNoise *43*

6.1.2 Parametri *46*

6.1.3 Calcolo della performance *46*

6.1.4 Dataset 1: "circ_1000_punti_noise_0.000" *47*

6.1.5 Dataset 2: "circ_1000_punti_noise_0.100" *50*

6.1.6 Dataset 3: "circ_1000_punti_noise_0.300" *52*

6.1.7 Dataset 4: "circ_5_punti_noise_0.000" *54*

6.1.8 Dataset 5: "circ_5_punti_noise_0.300" *56*
6.1.9 Dataset 6: "slice bunnyhole" *58*

7 ESPERIMENTI IN $\mathfrak{R}^3$ **61**

7.1 PCREC3 61
7.1.1 Dataset per le prove in 3D *61*
7.1.2 Parametri *63*
7.1.3 Dataset "venus" *64*
7.1.4 Dataset "bunnyhole" *66*
7.1.5 Dataset "knothole" *68*

8 CONCLUSIONI **70**

BIBLIOGRAFIA **71**

RINGRAZIAMENTI **72**

SOMMARIO

In questo lavoro di tesi di laurea specialistica viene affrontato il problema della ricostruzione di oggetti tridimensionali a partire da punti acquisiti mediante scanner. Queste nuvole di punti sono generalmente disorganizzate, affette da rumore e sovente presentano buchi di mancata acquisizione. Questo fa si che anche partendo dalla scansione di un oggetto 3D topologicamente equivalente ad una sfera, si possa ottenere una rappresentazione rumorosa e che presenta dei buchi.

Diversamente dagli approcci utilizzati tradizionalmente in questo campo, essenzialmente basati su *algoritmi di meshing*, nel presente lavoro si è seguito un approccio più innovativo introdotto alla fine degli anni novanta, basato sull'utilizzo delle reti neurali artificiali. Più precisamente, si è provato (con successo) a ricostruire la superficie utilizzando sistemi a regole fuzzy. Questo innovativo risultato, non ancora presente in letteratura, è stato possibile grazie all'uso di alcuni accorgimenti (fra cui un nuovo metodo di inferenza fuzzy) che ha reso il metodo efficiente sia in termini di complessità computazionale che di occupazione di memoria. Abbiamo chiamato questa nuova tecnica di inferenza *Truncated Fuzzy Inference*, ad indicare che l'inferenza coinvolge solo le *k* regole che si attivano di più.

Nella parte sperimentale vengono mostrati i risultati dell'applicazione di questa tecnica ad alcuni benchmark di visualizzazione utilizzati nella computer graphics.

In conclusione questo lavoro ha permesso di familiarizzare con i concetti di ricostruzione di superfici, insieme ad elementi di computer graphics, ed all'approfondimento dei sistemi a regole fuzzy come tecniche di approssimazione di funzioni.

1 INTRODUZIONE

Un modello digitale tridimensionale (3D) è una rappresentazione numerica di un oggetto reale. È possibile classificare i modelli in due grandi classi, cioè in *volumetric models* (vengono rappresentate le caratteristiche interne all'oggetto) e in *surface models* (viene rappresentata solo la superficie). In questo elaborato si farà riferimento principalmente ai secondi. Questi modelli possono essere visualizzati su uno schermo di computer come un'immagine 2D visibile da qualsiasi punto di vista utilizzando la proiezione prospettica e il rendering[1] dell'immagine. Il modello può anche essere inviato a un modulo di elaborazione per un'ulteriore elaborazione, come ad esempio nel campo del *Computerized Machinery Aided*, dove il modello è utilizzato per produrre automaticamente dal modello digitale una copia fisica dell'oggetto reale, processo solitamente indicato come il *reverse engineering*.

Le applicazioni basate sull'elaborazione di modelli tridimensionali sono sempre più popolari a causa della crescente disponibilità di dispositivi grafici tridimensionali e della diminuzione del costo della potenza di calcolo. La popolarità dei modelli digitali 3D deriva dal fatto che è possibile processarli in modo digitale: tra i vantaggi principali dei modelli digitali rispetto alle copie fisiche ci sono la loro facile visualizzazione e modifica e la possibilità di vedere in tempo reale l'interazione tra il modello e altri oggetti digitali. Vi sono alcune tipiche applicazioni che godono dell'uso di modellazione 3D come ad esempio:

- Archeologia, architettura e scultura: da un lato c'è il bisogno di conservare le opere d'arte e dall'altro c'è la possibilità di raggiungere un maggior numero di persone per promozione culturale. I musei virtuali consentono l'accesso alle opere d'arte ad un pubblico più grande rispetto ai musei reali, senza alcun rischio per gli oggetti esposti e, allo stesso tempo, possono promuovere il vero museo e attrarre visitatori. Le persone interessate a una singola opera d'arte hanno la possibilità di esplorare direttamente la sua rappresentazione virtuale che può essere accompagnata da informazioni multimediali. Inoltre i modelli

[1] Con *rendering* si identifica il processo di "resa" ovvero di generazione di un'immagine a partire da una descrizione matematica di una scena tridimensionale interpretata da algoritmi che definiscono il colore di ogni punto dell'immagine digitale

possono essere utilizzati per studiare gli edifici e verificarne anche la stabilità in ogni fase della costruzione.

- Fashion design, produzione e marketing: si può sfruttare l'uso di modelli 3D del corpo umano o delle sue parti. Nella moda virtuale, un cliente può creare il suo modello personale e fargli indossare i capi per valutare correttamente la vestibilità. In più permette di personalizzare i capi tenendo in considerazione le peculiarità del modello.
- Comparazione e classificazione di oggetti utilizzando una valutazione quantitativa delle caratteristiche del modello. Ad esempio nell'industria manifatturiera è utilizzata per il controllo qualità e nella sicurezza per identificare una persona tramite le caratteristiche biometriche.
- In applicazioni mediche: il modello 3D di parti del corpo e organi è in grado di offrire ai medici una vista virtuale del corpo umano per l'osservazione, come supporto alle diagnosi e/o come monitoraggio di una terapia.
- Formazione virtuale: per compiere correttamente le attività critiche o le procedure pericolose in cui l'errore umano deve essere ridotto al minimo. Un esempio è la chirurgia virtuale: i medici possono fare pratica su pazienti virtuali acquisendo maggior esperienza e fiducia prima di eseguire un intervento chirurgico vero e proprio. Un altro esempio è la simulazione di volo, utilizzata sia per la formazione sia per la valutazione dei piloti.
- Progettazione architettonica e reverse engineering. I modelli 3D possono essere utilizzati come un potente strumento per mostrare e discutere un progetto o un'idea. Sono sempre più utilizzati per progettare parti meccaniche ottimizzando la loro interazione con l'ambiente attraverso la visualizzazione dell'interazione in tempo reale. Questo è chiamato *visual computing*. I modelli 3D sono sempre stati considerati un valido strumento da utilizzare come input alla produzione in diversi campi. Più di recente, sono disponibili dei dispositivi creati per produrre direttamente un oggetto reale 3D da un modello CAD 3D, chiamati stampanti 3D.
- Industria dell'intrattenimento: utilizza sempre più spesso la modellizzazione 3D, infatti, negli ultimi anni il numero di film con personaggi digitali in 3D è aumentato notevolmente. L'uso di un modello 3D di un attore permette di evitare complesse, costose e lunghe sessioni di trucco per creare particolari caratteristiche fisiche, come pure l'uso di stant-men in scene pericolose. Analogamente la precisione e la raffinatezza dei numerosi moderni video

giochi 3D si basano proprio su un ampio uso di modelli 3D per creare scene e personaggi.

Applicazioni diverse possono avere esigenze diverse. Per esempio la ricostruzione in archeologia virtuale ha bisogno di una buona precisione e una bassa invasività, ma generalmente il tempo di scansione non è un vincolo importante. Al contrario, nelle applicazioni di videoconferenza, l'elaborazione in tempo reale è obbligatoria, mentre la qualità di modellazione gioca un ruolo secondario. Diversamente, nel controllo di qualità industriale è importante avere una ricostruzione veloce a un costo basso, in quanto le stesse operazioni vengono ripetute per molti oggetti dello stesso tipo.

Semplici oggetti possono essere rappresentati da modelli costituiti da singole equazioni. La Constructive Solid Geometry (CSG) è stata introdotta per creare oggetti più complessi dalla combinazione di oggetti solidi semplici tramite gli operatori di unione, di intersezione e di differenza. Sfortunatamente questo metodo non è in grado di rappresentare un grosso insieme di oggetti reali più complessi ed è quindi necessario ricorrere ai modelli Computer Aided Design (CAD). Gli oggetti 3D complessi possono essere creati in due modi diversi: disegnando le superfici utilizzando funzionalità avanzate di sistemi CAD o misurando direttamente la superficie dell'oggetto reale attraverso la scansione. Attualmente, gli operatori dei sistemi CAD sono in grado di creare modelli molto complessi utilizzando due principali tipi di strumenti: *NURBS* e *subdivision surfaces*. Le *Non-Uniform Rational B-Splines* (NURBS) sono un modello matematico che permette di generare curve e superfici con grande flessibilità e precisione. Le NURBS sono adatte per la gestione sia di figure che hanno una forma analitica sia di figure in forma libera. L'aspetto locale della curva è definito intuitivamente inserendo e spostando un insieme di punti che hanno un'influenza locale, chiamati punti di controllo. Le NURBS costituiscono quindi un quadro interattivo per creare modelli. Invece la *subdivision surfaces* è una tecnica per creare superfici lisce come risultato di un processo iterativo che parte dalla definizione grossolana di una mesh poligonale. La superficie liscia viene calcolata in modo ricorsivo partizionando la mesh poligonale iniziale: ad ogni passo i poligoni della mesh vengono suddivisi in nuovi poligoni con l'aggiunta di vertici che possono trovarsi al di fuori della superficie originale. A seconda della regola di suddivisione, la nuova mesh può implementare una versione più smussata delle mesh del passo precedente. La scansione è un processo che permette di ottenere il modello 3D in modo completamente automatico o semiautomatico misurando le caratteristiche geometriche dell'oggetto così come il suo aspetto (ad esempio, il colore e la struttura) e

individuando poi la superficie più appropriata al modello digitale che rappresenta i dati rilevati.

Nei sistemi CAD l'operatore umano plasma interattivamente il modello superficiale per rappresentare un dato oggetto, mentre gli scanner 3D sono finalizzati a produrre la rappresentazione digitale della superficie automaticamente. Gli scanner sono generalmente più veloci rispetto all'utilizzo di un sistema CAD e raggiungono un più alto livello di precisione dell'oggetto scannerizzato. Inoltre, la scansione, essendo essenzialmente un processo di misura, non richiede alcuna abilità artistica dell'operatore. D'altra parte, la creazione di un modello digitale attraverso la scansione richiede che sia il modello sia il sistema di misura siano disponibili.

1.1 Scansione di oggetti 3D

Il processo di acquisizione della superficie di un oggetto reale si chiama scansione 3D e lo strumento usato a questo scopo è chiamato scanner 3D. Gli scanner 3D possono avere caratteristiche molto diverse, perché le loro applicazioni possono essere di vario genere. Sono caratterizzati da differenti prestazioni e le tecniche di acquisizione utilizzate si basano su principi fisici diversi. Tuttavia, nonostante le differenze, tutti gli scanner hanno una pipeline comune per costruire un modello digitale 3D. Infatti, la costruzione del modello 3D è un processo modulare composto da diverse fasi. I sensori utilizzati negli scanner 3D campionano l'oggetto in un insieme di punti acquisendo quindi un insieme finito di informazioni, come le coordinate 3D di punti appartenenti alla superficie dell'oggetto, campi di colore e texture. Partendo da queste informazioni uno scanner 3D, attraverso elaborazioni adeguate, ottiene una descrizione continua della superficie dell'oggetto. Questo processo, in termini statistici, è noto anche come generalizzazione. Inoltre, poiché l'acquisizione è realizzata per mezzo di una sequenza di misure, le caratteristiche misurate sono affette da errori di misura che dovranno essere rimosse dalla superficie ricostruita. Quindi la procedura di ricostruzione deve considerare delle strategie per il filtraggio dei dati rumorosi.

I passi che esegue lo scanner per la costruzione di un modello 3D sono (Figura 1.1):

1. Una serie di sensori viene utilizzata per catturare dati sulla superficie e sul suo colore (fase di digitalizzazione).

2. Da questi dati viene calcolata una generalizzazione dei dati acquisiti su tutta la superficie dell'oggetto (fase di generalizzazione).
3. Vengono unite le informazioni provenienti da sensori diversi o da diverse sessioni di acquisizione (fase di integrazione).
4. I passi precedenti possono essere iterati per migliorare la qualità di modello (ad esempio, per ottenere un insieme denso di dati in alcune parti dell'oggetto).
5. La rappresentazione digitale 3D viene trasformata e ottimizzata utilizzando il miglior paradigma per una specifica applicazione (fase di ottimizzazione).

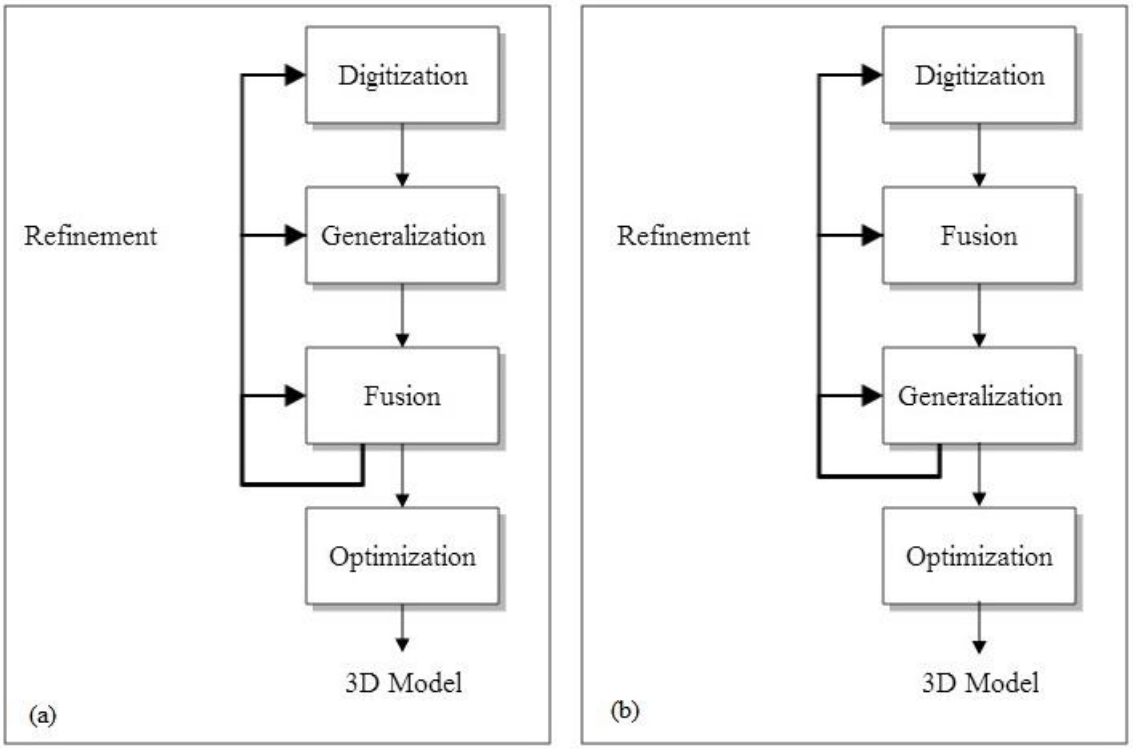


Figura 1.1 - *Lo schema mostra le principali fasi del processo di digitalizzazione di oggetti 3D. Nel grafico (a) viene prima costruito un modello dell'oggetto per ogni set di dati acquisiti, i modelli ottenuti sono fusi nella fase successiva. Nel grafico (b) avviene direttamente la fusione sui punti dai dati grezzi e si ottiene poi un singolo modello dall'intero insieme di punti.*

Ciascuno dei punti precedenti può essere migliorato sfruttando una conoscenza a priori sulle tecniche di acquisizione o sull'oggetto da esaminare. Sfruttando la conoscenza a priori è possibile migliorare il modello 3D finale sia in termini di qualità sia di complessità computazionale. Le tecniche sviluppate per i casi generali possono essere migliorate e personalizzate per oggetti più particolari.

1.1.1 Digitalizzazione

Generalmente, il primo passo nella creazione di un modello 3D consiste nel catturare le informazioni geometriche e il colore dell'oggetto reale. Gli oggetti possono essere piccoli come monete o grandi come edifici, possono essere fermi o muoversi durante la scansione, e questo ha portato allo sviluppo di tecnologie e strumenti molto diversi. Fondamentalmente questi sistemi funzionano misurando il riflesso generato da un certo tipo di radiazione quando viene a contatto con la superficie dell'oggetto. Il risultato di questa fase è generalmente un insieme di punti 3D campionati sulla superficie dell'oggetto, a volte arricchiti con un colore per la superficie e con informazioni di riflessione. L'insieme dei punti 3D è spesso denominato *points cloud*, nuvola di punti.

1.1.1.1 Tipi Di Scanner 3D

Uno scanner 3D è composto da un insieme di dispositivi e procedure capaci di catturare la forma e l'aspetto di un oggetto. In generale, gli scanner sono basati sul campionamento della superficie. I dati raccolti sono utilizzati nella fase di ricostruzione per ottenere il modello digitale dell'oggetto. Un sistema scanner 3D è, sostanzialmente, uno strumento di misura e quindi può essere valutato in termini di accuratezza, risoluzione, velocità, flessibilità, robustezza, usabilità, invasività e costo.

Alcune di queste proprietà sono facilmente quantificabili, mentre altre possono essere valutate solo qualitativamente. Inoltre, alcune proprietà dipendono dai componenti del sistema. Per esempio, il principio fisico utilizzato per ottenere la misurazione determina l'invasività del sistema. Alcune funzioni possono essere incompatibili; per esempio un alto grado di usabilità non può coesistere con un elevato livello di flessibilità e con un alto livello di precisione, in quanto entrambe le ultime due proprietà generalmente richiedono un sistema complesso, in cui sono necessari utenti con un grado significativo di competenza. Il principio fisico sfruttato dal sistema per la misurazione delle caratteristiche geometriche dell'oggetto è probabilmente la caratteristica più importante di uno scanner 3D, e può essere utilizzato per classificare i diversi sistemi. I sistemi di scansione 3D utilizzano, principalmente, due famiglie di tecniche di acquisizione. La prima è basata sull'interazione tra un sensore e la superficie dell'oggetto; i sistemi che utilizzano questo approccio sono chiamati *Contact 3D scanner*. La seconda è basata sull'interazione tra una

radiazione (elettromagnetica o audio) e la superficie dell'oggetto; in questo caso i sistemi sono chiamati *Non-contact 3D scanner*.

Tra i Contact 3D scanner si differenziano principalmente due tipi: *Coordinate Measuring Machines* (CMM) and *Jointed Arm systems*. Il primo è costituito da una sonda tattile attaccata ad un braccio verticale, che può essere spostato lungo il piano orizzontale. Il movimento è prodotto dalla traslazione lungo i tre assi ortogonali mentre la misurazione è il risultato diretto dello spostamento dell'azionatore lungo ciascun asse. L'oggetto deve essere disposto sopra una piastra di riferimento in modo che la sonda possa esplorarlo. Il secondo invece è composto da una catena di collegamenti snodati con una sonda all'estremità.

Nei Non-contact 3D scanner il campionamento della superficie viene eseguito dall'interazione tra un certo tipo di radiazione e la superficie dell'oggetto stesso. Questi sistemi possono essere suddivisi in due sottocategorie: sistema trasmissivo se la radiazione passa attraverso l'oggetto, sistema riflettente se viene riflessa dalla superficie dell'oggetto.

1.1.2 Generalizzazione

La ricostruzione di una superficie da una nuvola di punti tridimensionali richiede la definizione implicita della topologia[2] della superficie. Qualsiasi procedura basata sulla prossimità può avere risultati sbagliati se invece della distanza topologica viene considerata la distanza geometrica. Per questo motivo una semplificazione al problema della ricostruzione può essere la conoscenza a priori della topologia della superficie dell'oggetto. Alternativamente, la mancanza di informazione topologica può essere gestita in due fasi. Nella prima fase viene determinato un modello poligonale che rappresenta i dati a bassa risoluzione. Nella seconda fase viene costruita la superficie, definendola come uno spostamento rispetto al modello poligonale calcolato nella prima fase, che quindi funge da base per la superficie 3D.

Il problema della ricostruzione di superficie da nuvole di punti è stato largamente studiato negli ultimi anni. Le tecniche di ricostruzione sono classificate in due classi

[2] La distanza topologica è definita come il percorso più breve che collega due punti, tutto contenuto all'interno della superficie mentre la distanza euclidea o geometrica è definita come il segmento di linea più breve che collega i due punti. Quindi distanze topologiche e geometriche possono essere molto diverse.

principali: *volumetric techniques*, che mirano a individuare il volume di spazio occupato dall'oggetto, *surface fitting techniques*, che hanno lo scopo di identificare la copia 2D associata alla superficie dell'oggetto, all'interno dello spazio 3D.

1.1.3 Integrazione di scansioni multiple

Quando un oggetto non è osservabile da un singolo sensore o è richiesto un elevato livello di dettaglio rispetto alla dimensione dell'oggetto, è necessario un ulteriore passaggio per integrare i dati *(data integration).* Infatti, in questo caso, la scansione deve essere eseguita da diversi punti di vista e le informazioni acquisite dalle diverse viste devono essere integrate al fine di calcolare una singola rappresentazione. Questa operazione può essere divisa in due fasi:

- la registrazione, in cui i diversi dati sono rappresentati nello stesso sistema di riferimento;
- la fusione, in cui viene generato un unico modello che combina i dati provenienti da diversi punti di vista.

La scansione da molti punti di vista può essere realizzata con l'acquisizione della superficie dell'oggetto nella stessa sessione, utilizzando più dispositivi di scansione ognuno dei quali finalizzato a una parte specifica dell'oggetto, oppure programmando più sessioni di acquisizione, in modo che le sessioni di acquisizione supplementari siano finalizzate ad acquisire dati solo nelle parti dell'oggetto in cui la qualità di ricostruzione è troppo povera.

Ogni volta che i dati, provenienti da diverse fonti di informazioni, devono essere correlati al medesimo sistema di riferimento, è necessario eseguire una fase di registrazione, anche quando bisogna unire diversi modelli dello stesso oggetto. In quest'ultimo caso, i modelli devono essere scalati e allineati parallelamente tra loro per permettere un corretto confronto. Anche dopo la registrazione, le superfici appartenenti a diversi punti di vista non possono sovrapporsi perfettamente, a causa degli errori di calibrazione e di ricostruzione che si aggiungono all'errore di misura. Inoltre le due superfici devono essere fuse in modo tale da non risultare più dense nella regione di sovrapposizione. La semplice unificazione dei dati registrati o delle superfici parziali produrrebbe una cattiva rappresentazione della superficie dell'oggetto, con artefatti nelle

regioni sovrapposte. L'obiettivo della procedura di fusione è di elaborare un unico modello, migliorando la rappresentazione della superficie nella regione di sovrapposizione.

1.1.4 Ottimizzazione

Un modello 3D può essere utilizzato in diverse applicazioni. A seconda delle caratteristiche dell'applicazione finale può essere necessaria una post-elaborazione del modello. Ad esempio, le applicazioni di realtà virtuale richiedono di solito modelli composti da un numero di poligoni il più piccoli possibile, mentre, se il modello deve essere utilizzato in un film, può essere necessario modificarlo per un migliore inserimento all'interno di una scena.

La trasformazione può essere semplificata con l'utilizzo di particolari paradigmi scelti per la rappresentazione del modello. Le procedure di lavorazione più comuni applicate a un modello sono:

- *Conversion*: è applicata per ottenere una rappresentazione delle informazioni in un formato adatto per la lavorazione in altre applicazioni;
- *Compression:* per la descrizione di modelli complessi, in generale si cerca di minimizzare la perdita di informazioni percettibile, data una certa quantità di risorse disponibili;
- *Watermarking*: è una procedura correlata alla compressione che introduce informazioni che non sono percepibili dall'utilizzatore del modello, ma che possono essere rilevate da un adeguato processo costituendo la firma del modello digitale.
- *Animation*: viene ottenuta associando una legge di moto (come funzione del tempo) per i punti della superficie;
- *Morphing*: è l'operazione che viene utilizzata per trasformare l’aspetto geometrico e visivo di un oggetto in quello di un altro oggetto.

2 RICOSTRUZIONE DI SUPERFICI

Il passo successivo al campionamento dei dati sulla superficie di un oggetto è la generalizzazione di tali dati per ottenere una descrizione continua della superficie dell'oggetto stesso, cui possono essere associati attributi visivi, come il colore, la trama e la riflessione. In questo capitolo viene presentata una panoramica delle tecniche utilizzate per la generalizzazione. Queste possono essere suddivise in due grandi famiglie: *volumetric method* e *surface fitting method.*

2.1 Costruzione della superficie da una nuvola di punti

L'obiettivo della fase di ricostruzione è trovare la miglior approssimazione della superficie per l'insieme dei dati raccolti nella fase di acquisizione. Ci sono due aspetti che caratterizzano questo problema: la conoscenza a priori dell'oggetto e il tipo di informazioni raccolte in fase di acquisizione. Per risolvere questo problema deve essere scelto un paradigma di rappresentazione della superficie in modo tale che possa essere codificata una conoscenza a priori del problema. Ad esempio, se l'acquisizione è limitata a una certa classe di oggetti, si può utilizzare un modello parametrico per rappresentare gli oggetti di quella classe. La ricostruzione della superficie diventa una ricerca dei parametri che meglio si adattano ai dati acquisiti. Se la conoscenza a priori dell'oggetto è limitata o la classe di oggetti è molto grande, il paradigma di ricostruzione deve avere molti parametri al fine di accogliere numerose forme differenti e la tecnica di ricostruzione dovrebbe essere più complessa. Il problema di ricostruzione della superficie può essere formulato come un problema di ottimizzazione. Sebbene un approccio basato su modello sia più robusto, può essere applicato in pochi casi. In questo capitolo viene esaminato il caso generale in cui la conoscenza a priori è fortemente limitata.

Per trasformare una nuvola di punti in una superficie continua sorgono due problemi: la superficie ha una descrizione continua, mentre il campionamento è una forma di discretizzazione (il valore della superficie tra i punti campionati deve essere stimato) e i campioni sono affetti da errore di misura (potrebbe essere necessaria una strategia per il filtraggio dell'errore).

L'obiettivo della ricostruzione della superficie è, quindi, il calcolo di un'approssimazione di una superficie sconosciuta utilizzando un insieme di punti ed eventualmente informazioni ausiliarie sulla superficie originale o riguardo il processo di campionamento (ad esempio, la densità del campione e la grandezza del rumore). Sono state proposte in letteratura due grandi famiglie di soluzioni, chiamate spesso *spatial subdivision techniques* e *surface fitting techniques*. Alcuni metodi producono una superficie ruvida usando tecniche di suddivisione spaziale e perfezionano la soluzione utilizzando approcci a surface fitting. Nel seguito saranno presentati i due approcci.

2.2 Tecniche di suddivisione dello spazio

Le tecniche di suddivisione dello spazio sono basate sulla ricerca del volume occupato dall'oggetto. Gli algoritmi di questa classe di tecniche sono composti dai seguenti passi:

1. decomposizione dello spazio dei punti nelle celle,
2. selezione delle celle attraversate dalla superficie,
3. computazione della superficie dalle celle selezionate.

Si riporta un esempio di meshatura triangolare in Figura 2.1, dove la bounding box dei punti viene regolarmene suddivisa in cubi (Fig.2.1a-2.1b). Sono considerati solo i cubi che hanno all'interno almeno un punto (Fig.2.1b). Una prima approssimazione della superficie è composta dalle facce esterne dei cubi che non sono condivise tra coppie di cubi. Queste facce sono poi divise dalla diagonale in due parti al fine di ottenere una mesh (maglia) triangolare (Fig.2.1c). Spostando ogni vertice della mesh nel punto che si ottiene facendo la media pesata della sua vecchia posizione e di quella dei suoi vicini, si ottiene una versione smussata della superficie (Fig.2.1d). La ricostruzione viene poi migliorata adattando la superficie risultante ai punti dati. Un fattore critico dei metodi volumetrici basati su un reticolo regolare è la dimensione delle celle.

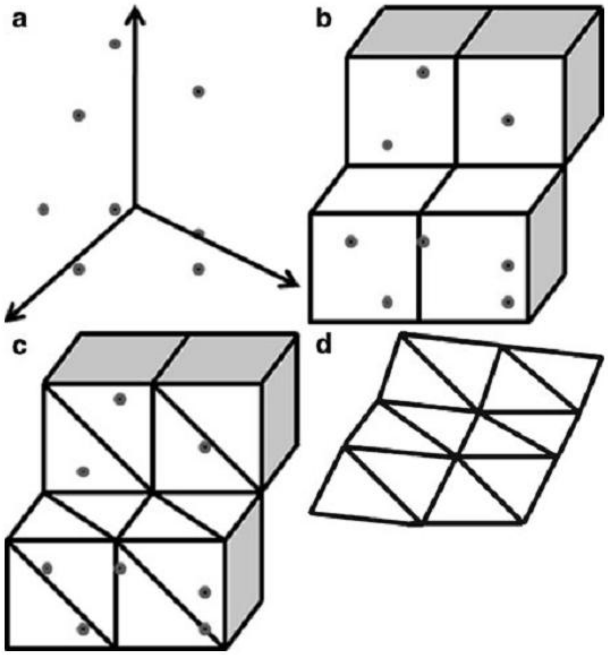


Figura 2.1 - *Esempio di ricostruzione della superficie con metodo volumetrico. Nel grafico (a) la nuvola di punti, nel grafico (b) la partizione in cubi dello spazio dei punti, nel grafico (c) la prima maglia triangolare, e nel grafico (d) un'anteprima della maglia smussata.*

Un altro tipo di mesh è quella in cui sono utilizzate delle celle tetragonali. Prima di tutto viene calcolata un'approssimazione della superficie usando le α-shape[3]. Per ogni punto viene calcolato uno scalare che rappresenta la distanza tra il punto e la superficie. L'algoritmo inizia con un singolo tetraedro che comprende tutti i punti, che è poi diviso in quattro tetraedri aggiungendo il baricentro ai quattro vertici; per ciascuno dei cinque punti viene calcolata la distanza dalla superficie approssimata. I tetraedri i cui vertici hanno lo stesso segno del loro scalare (tutti positivi o tutti negativi) vengono eliminati. Per ciascuno dei restanti tetraedri viene calcolata un'approssimazione della superficie che lo attraversa. Se l'errore di approssimazione della superficie rispetto ai punti interni al tetraedro è oltre una certa soglia, la procedura viene ripetuta.

Un altro metodo prevede che la nuvola di punti venga partizionata mediante l'algoritmo *k-means clustering*. Ciascun cluster è rappresentato da un'approssimazione poligonale e il modello ottenuto è utilizzato come prima approssimazione della superficie. Per evitare il problema di mettere negli stessi cluster punti che appartengono a facce

[3] α-shape è una generalizzazione della triangolazione di Delaunay. The α-shape di un insieme di punti P si ottiene dalla sua triangolazione di Delaunay, rimuovendo gli elementi (spigoli, triangoli e tetraedri) che non possono essere inscritti in una sfera di raggio α.

diverse di un oggetto, vengono utilizzate le normali ai punti (stimate dalle proprietà locali dell'insieme di dati).

Ci sono metodi in cui viene utilizzata una *mesh deforming procedure*. Il modello iniziale è un poliedro non autointersecante che è incorporato nell'oggetto o lo circonda nella rappresentazione del volume dei dati. Poi il modello viene deformato rispetto ad un insieme di vincoli. Per ogni vertice viene associata una funzione che descrive il costo delle deformazioni locali; la funzione considera il rumore dei dati, le caratteristiche di ricostruzione e la semplicità della mesh.

L'adattamento iterativo dei parametri di superficie è stato ampiamente studiato nel campo delle connessioni in cui il problema viene visto come un caso particolare di apprendimento. Il modello Self-Organizing Map (SOM) è costituito da un insieme di unità, $\{u_j\}$, organizzate in una struttura reticolare e caratterizzate da un valore, $w_j \in \Re^D$ che rappresenta la posizione dell'unità nello spazio dei dati. La SOM è configurata con lo scopo di approssimare la distribuzione di un dato insieme, $P \in \Re^D$. Per ogni passo dell'adattamento viene estratto un elemento dall'insieme di dati p_i e l'unità più vicina nello spazio dei dati, u_k, chiamata unità vincente, e le unità di collegamento vengono adattate con la seguente regola:

$$w_j(i+1) = w_j(i) + \eta \cdot h_j(k) \cdot (p - w_j(i))$$

Il parametro η è il tasso di apprendimento e controlla la velocità di adattamento, mentre il valore assunto da $h_j(k)$ è inversamente proporzionale alla distanza tra le unità *j* e *k*, valutata sulla struttura reticolare (distanza topologica).

2.3 Metodi di Surface fitting

L'alternativa alle tecniche di suddivisione spaziale è rappresentata dal *surface fitting*. Queste tecniche si dividono in due classi, definite *surface reconstruction* o *function reconstruction*. La prima può essere ulteriormente suddivisa in altre due classi: *implicit reconstruction* e *parametric reconstruction*. Nella ricostruzione implicita l'obiettivo è di trovare una funzione di smussamento $f : \Re^3 \to \Re$ tale che i punti d'ingresso $\{k_1, ..., k_n\}$

siano vicini ai punti in cui $f(\bullet)$ è zero, $Z(f)=\{x\in\Re^3 \mid f(x)=0\}$. $Z(f)$ rappresenta la stima della superficie. In una seconda fase, può essere utilizzato un algoritmo che crea il contorno per ottenere una semplice superficie che approssima $Z(f)$. Nella ricostruzione parametrica, invece, la superficie è rappresentata da una funzione parametrica sul dominio bidimensionale dei parametri incorporato in $\Re^3$. Per esempio una formulazione parametrica molto diffusa è il Non-Uniform Rational B-spline (NURBS), che è definito come:

$$f(u,v)=\sum_i\sum_j B_{i,j}^h N_{i,k}(u) M_{i,l}(v)$$

dove $N_{i,k}(u)$ e $M_{i,l}(v)$ sono le funzioni B-spline base di ordine *k* e *l*, $B_{i,j}^h$ sono le coordinate dei punti di controllo. Questo approccio ha un limite: in generale, è molto difficile creare una superficie definita su punti di controllo da un insieme disorganizzato di punti acquisititi da scanner. Per questi casi, in pratica, la ricostruzione parametrica è usata solo per una descrizione approssimativa del modello iniziale che deve essere poi migliorato con altre tecniche. D'altra parte questo approccio è particolarmente adatto per la modellazione interattiva perché spostando i punti di controllo è facile controllare la deformazione del modello della superficie.

L'obiettivo del metodo *function reconstruction* può essere indicato come segue: D*ato un insieme di coppie* $\{(x_1,z_1),...,(x_n,z_n)\}$*, dove* $x_i\in\Re^2$ *e* $z_i\in\Re$*, l'obiettivo è di determinare una funzione* $f:\Re^2\to\Re$ *tale che* $f(x_i)\approx z_i$*. Più formalmente* $f(x_i)=z_i+\varepsilon_i$ *dove* ε_i *è una variabile casuale con una data distribuzione.*

L'approccio a ricostruzione implicita permette la completa ricostruzione 3D, ma la superficie è descritta in termini di una funzione di distanza. Il dominio degli input è rappresentato dalle coordinate 3D dei punti, il codominio è una misura della distanza tra i punti e la superficie è implicitamente calcolata dalla funzione stessa. Questo significa che i punti appartenenti alla superficie devono essere stimati ricercando gli elementi nell'insieme degli zeri. Questo è in genere più costoso rispetto al caso di ricostruzione della funzione, in cui la funzione stessa rappresenta direttamente la superficie. Inoltre, avendo la rappresentazione analitica della superficie, si può facilmente ottenere un ricampionamento della stessa ad una risoluzione specificatamente richiesta. Con un approccio a ricostruzione della funzione non può essere realizzata la descrizione di un

modello completo in 3D utilizzando una singola funzione esplicita, poiché è in grado di descrivere solo una superficie in 2.5D (cioè simile a una rappresentazione in bassorilievo). È quindi necessario un ulteriore passo per la registrazione e la fusione delle diverse ricostruzioni 2.5D.

Nell'approccio a ricostruzione implicita viene usato il concetto di *signed distance function* per l'approssimazione della superficie implicita. Questa funzione calcola per ciascun punto nello spazio 3D la distanza segnata dal punto alla superficie lungo la normale alla superficie del punto stesso (per questo calcolo è richiesta quindi la stima delle normali alla superficie). La funzione di distanza con segno della superficie assume un valore positivo per i punti esterni e valore negativo per punti interni. L'isosuperficie (cioè la regione dello spazio in cui la funzione di distanza con segno è zero) è poi la superficie cercata. Lo spazio 3D occupato dalla nuvola di punti viene partizionato in modo regolare in una griglia di voxel[4]. Per ciascun vertice voxel viene stimato il valore della funzione distanza con segno considerando la distanza tra il vertice e il piano che approssima i punti in prossimità del vertice. Poi vengono selezionati i voxel che hanno i vertici di segno opposto, poiché sono i voxel attraverso cui passa la superficie. Per migliorare la risoluzione della superficie viene applicato ai cubi selezionati il Marching cubes. Questo algoritmo è uno degli algoritmi più importanti per il calcolo dell'isosuperficie dalle funzioni di distanza con segno. Ogni cubo selezionato ha alcuni vertici con segno della funzione sopra lo zero e altri sotto lo zero, l'iso-superficie di valore uguale a zero passerà attraverso il cubo. Considerando il valore del segno della funzione nei vertici vengono calcolate le patch triangolari che dividono il cubo in regioni interne ed esterne all'isosuperficie. La rappresentazione della superficie viene generata collegando i patch triangolari ottenuti.

Un approccio simile è quello basato su una tecnica differente per il calcolo della funzione di distanza con segno. In questo caso il valore della funzione distanza con segno nei vertici del voxel viene calcolato come media ponderata della distanza tra ogni punto negli otto voxel vicini e il vertice considerato. Questo considerando il caso in cui i punti vengono acquisiti utilizzando uno scanner laser, l'errore di misura è in gran parte causato dall'angolo tra il punto normale e la linea di vista dello scanner 3D. In particolare, l'errore di misurazione sarà più grande quando la normale alla superficie è vicina a essere ortogonale alla linea di vista dello scanner 3D. Per questa ragione viene usato il coseno di

[4] Un voxel (*volumetric pixel* o più precisamente *volumetric picture element*) è un elemento di volume che rappresenta un valore d'intensità di segnale o di colore in uno spazio tridimensionale, analogamente al pixel che rappresenta un dato di un'immagine bidimensionale

quest’angolo per pesare il contributo di ogni punto. Poiché la funzione di distanza determina implicitamente la superficie e ovviamente non è nota a priori, la tecnica per stimare la funzione di distanza è un aspetto critico per i metodi di questo tipo.

3 IL SURFACE FITTING COME PROBLEMA DI REGRESSIONE

Nei capitoli precedenti è stata fatta una breve panoramica sui metodi per la ricostruzione delle superfici. In questo capitolo l'attenzione sarà focalizzata su una particolare classe di metodi che vedono la ricostruzione della superficie come un problema di approssimazione multivariata. Saranno presentate alcune delle tecniche più popolari di questo tipo e saranno discussi i pro e i contro.

3.1 Problemi di regressione

La previsione del valore di una variabile dalle sue osservazioni è nota in letteratura come regressione statistica. Questo problema è stato studiato in varie discipline nell'ambito dell'informatica e della matematica applicata. In particolare la previsione del valore reale di una variabile è un tema importante per il *machine learning*, l'apprendimento automatico. Quando il valore da prevedere è nominale o discreto, il problema è chiamato classificazione. Anche se nell'ambito dell'apprendimento automatico la maggior parte del lavoro di ricerca si è concentrata sulla classificazione, negli ultimi dieci anni l'attenzione si è spostata verso la regressione, poiché un gran numero di problemi della vita reale può essere modellato come un problema di regressione. Il problema di regressione è identificato con diversi nomi, per esempio: *functional prediction* (previsione funzionale), *real value prediction* (la previsione di valori reali), *function approximation* (approssimazione di funzioni) *continuous class learning* (l'apprendimento di classi continuo). Più formalmente, un problema di regressione può essere definito come segue:

Dato un insieme di campioni $\{(x_i, y_i) \mid i = 1, ..., n\}$, *dove* $x_i \in X \subset \Re^D$ *e* $y_i \in Y \subset \Re$, *l'obiettivo è di stimare una funzione incognita f, detta funzione di regressione, tale che* $y = f(x) + \varepsilon$, *dove* ε *è una variabile casuale (a media zero) che modella il rumore sul dataset.*

Gli elementi del vettore *x* saranno indicati come variabili di input (o attributi) e quelli del vettore *y* come variabili di uscita (o target). La funzione di regressione descrive

la dipendenza tra *x* e *y*. L'obiettivo di un problema di regressione è stimare questa dipendenza da un insieme di campioni ed eventualmente da una conoscenza a priori sulla funzione di regressione *f*. La soluzione di un problema di regressione verrà indicata con $\hat{f}$. Come $\hat{f}$ può essere valutata anche per valori di *x* che non sono nel training set, così la funzione di regressione rappresenta una generalizzazione del training set. La generalizzazione di una soluzione può essere valutata sfidando $\hat{f}$ su un insieme di campioni (generalmente chiamati test set) non utilizzati nel calcolo di $\hat{f}$.

Si noti che il problema di regressione è più generale dei problemi in cui i campioni di training sono considerati senza rumore (ad esempio problemi di interpolazione). Tipicamente, le tecniche sviluppate per dati affetti da rumore possono fornire buone soluzioni anche nei casi in cui il rumore è assente. Viceversa le tecniche sviluppate per il caso senza rumore non sono in grado di filtrarlo e la soluzione tende a riprodurre anche il rumore sui campioni di training. In questo caso la soluzione non generalizza bene. Quando $\hat{f}$ è caratterizzata da un basso errore sui campioni di training e di elevato errore sugli altri dati, si è verificato un *overfitting*. Le tecniche di regressione possono essere considerate robuste perché mirano al filtraggio del rumore e ad evitare overfitting.

L'importanza delle tecniche di regressione è evidenziata in tutte le applicazioni in cui non è disponibile un esperto del dominio. Grazie a queste tecniche, il problema può essere affrontato utilizzando solo i dati. Inoltre, può essere utilizzata una tecnica per estrarre automaticamente la conoscenza. Infatti, come la soluzione permette di predire l'uscita corrispondente ai valori in ingresso non utilizzati nella stima di $\hat{f}$, essa può essere usata per ottenere nuove conoscenze sul dominio considerato. Le tecniche di regressione possono essere suddivisi in due classi: parametriche e non parametriche.

3.2 Approcci parametrici vs approcci non parametrici

Uno degli approcci più utilizzati nella regressione è di limitare la ricerca della soluzione a un sottoinsieme di funzioni. In particolare la limitazione può essere realizzata ricercando la soluzione tra una famiglia di funzioni parametriche. Sebbene la scelta di una piccola famiglia di funzioni parametriche possa avere l'effetto collaterale di escludere la funzione che genera effettivamente il dataset (che sarebbe la "vera" soluzione del problema

di regressione), questo approccio offre il vantaggio di una soluzione più semplice (ad esempio è computazionalmente meno costoso rispetto ad altri metodi). La regressione lineare semplice nell'analisi statistica è un esempio di un approccio parametrico. In questo approccio, la variabile *y* dovrebbe cambiare a un tasso costante rispetto alla variabile *x*. Quindi la soluzione del problema di regressione sarà un iperpiano che soddisfa il seguente sistema lineare:

$$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + ... + \beta_D x_{i,D} + \varepsilon_i,\ i = 1, \ldots, n$$

dove il primo indice *i* denota le osservazioni o le istanze, mentre il secondo indica il numero di variabili d'ingresso. Quindi può essere utilizzato un sistema di equazioni lineari *n* per calcolare D+1 parametri, β_j , *j* = 0, …, D dell'iperpiano. Il termine ε_i rappresenta l'errore della soluzione sul campione *i*, ossia $\varepsilon_i = y_i - \hat{y}_i$ dove $\hat{y}_i = \hat{f}(x_i)$, il valore della soluzione (l'iperpiano) per x_i. In generale, il criterio utilizzato è basato sulla minimizzazione di una funzione di errore (o perdita) sui punti del training. Una misura dell'errore utilizzata spesso nelle applicazioni è la somma della differenza dei quadrati tra il valore predetto e il valore effettivo degli esempi (*least squares criterion*).

Nei modelli parametrici viene data la struttura matematica di $\hat{f}$ ed è prevista per essere adatta ai dati. Questo permette di ridurre ampliamente il numero dei parametri β_j al prezzo di una forma più complessa di $\hat{f}$. Questo approccio è ragionevole solo quando è disponibile un'adeguata informazione a priori sui dati. Quando la conoscenza a priori è scarsa o non disponibile, è generalmente preferibile un approccio non parametrico.

L'approccio non parametrico si basa sull'impiego di una struttura generica di $\hat{f}$ in grado di modellare un gran numero di funzioni, che richiedono un'ipotesi molto debole sulla distribuzione dei dati sottostanti. Un esempio di approccio non parametrico è la regressione localmente ponderata (*locally weighted regression)*.

La scelta del possibile insieme di funzioni tra cui la soluzione deve essere cercata è nota come problema di selezione del modello. È un compito critico a causa della *bias-variance dilemma*. Si può dimostrare che il valore di aspettazione dell'errore commesso da un modello, *M*, per la soluzione di un problema di regressione può essere espresso come somma di tre componenti: $\text{Var}\left(\text{M}\right) + \text{Bias}\left(M\right) + \sigma_\varepsilon$.

Il primo termine, Var (M), descrive la robustezza di un modello rispetto al training set. Se diversi training set campionati dalla stessa distribuzione determinano soluzioni molto differenti, allora il modello M è caratterizzato da una grande varianza. Il secondo termine, Bias (M), descrive come la migliore approssimazione ottenibile dal modello M, è una buona stima della funzione di regressione *f*. Il terzo termine σ_ε descrive l'errore di misura sui dati e rappresenta il limite della precisione della soluzione.

Ad esempio, se la funzione di regressione è un polinomio quadratico e il problema viene risolto utilizzando un modello lineare di regressione, la varianza può essere piccola (la soluzione può essere robusta rispetto a diversi insiemi di training), ma la distorsione può essere grande (la miglior soluzione per il modello non è una buona stima della funzione di regressione). Al contrario se il modello è non lineare, la varianza può essere grande (come la soluzione tende a overfittare i dati e cambia per diversi training set), ma la distorsione può essere piccola (il modello è in grado di riprodurre un polinomio quadratico, che è una possibile buona stima della funzione di regressione). In generale, un buon modello realizza un compromesso tra distorsione e varianza. La soluzione può essere calcolata sia con approcci parametrici sia con approcci non parametrici, principalmente utilizzando due classi di metodi: metodi locali, che cercano localmente la soluzione sulla media dei dati, e metodi globali, in cui la soluzione è trovata risolvendo un problema di ottimizzazione globale.

3.3 Regressione basata su istanze (Instance-based regression)

Gli approcci basati su istanze formano una famiglia di metodi non parametrici che invece di configurare un modello di generalizzazione esplicito, utilizzano una strategia per la predizione basata solo sul confronto diretto di nuove istanze con quelle contenute nel training set. In altre parole, i parametri del modello sono esattamente le istanze training set, e l'elaborazione dei dati di training viene effettuata nel momento in cui la predizione deve essere eseguita. Per questo tali metodi sono anche chiamati algoritmi *lazy learning*, di apprendimento "pigri". Questa classe di algoritmi deriva dal *nearest neighbor classifier*, un classificatore dei vicini più vicini, che è il più semplice membro di questa famiglia. Questi metodi sono stati inizialmente dedicati alla classificazione e sono stati estesi in seguito ai casi di regressione.

Nelle tecniche di regressione basata su istanze gli esempi sono costituiti da un insieme di variabili di input, il cui valore può essere nominale o numerico, e dalle variabili di uscita che sono continue. Il valore previsto per l'etichetta di un'istanza è calcolato in funzione degli elementi del training set. Ad esempio, il metodo nearest neighbor considera i campioni di training (detti vicini) più simili alla nuova istanza, e calcola il valore della variabile della nuova istanza come funzione (tipicamente la media) delle variabili di uscita del sottoinsieme dei suoi vicini. La somiglianza può essere calcolata considerando la distanza euclidea tra istanze. In letteratura esistono diverse varianti degli algoritmi basati su istanza. Il più semplice è chiamato *proximity algorithm*, algoritmo di prossimità. La somiglianza è calcolata come il complemento della distanza euclidea, $S\left(x_i, x_j\right) = 1 - \left\| x_i - x_j \right\|_1$, dove tutte le variabili d'ingresso sono state normalizzate nell'intervallo continuo [0,1]. Il valore della grandezza in uscita per un dato campione *x* viene calcolato come la somma pesata dei valori delle variabili d'uscita simili agli elementi del training set:

$$\hat{f}\left(x\right) = \frac{\sum_{x_j \in T_i} S\left(x, x_j\right) y_j}{\sum_{x_j \in T_i} S\left(x, x_j\right)}$$

dove T_i è un sottoinsieme del training set composto da elementi del training set che hanno maggior somiglianza con *x*. Questo metodo esegue una buona approssimazione solo per un training set sufficientemente grande e quando *f* può essere linearizzata localmente. Il metodo si basa sul presupposto che tutte le variabili di input hanno la stessa importanza rispetto alla variabile di uscita. Se questa ipotesi non è verificata, il metodo deve essere modificato applicando un peso ad ogni variabile di input. Dal momento che tutti gli esempi di training devono essere memorizzati, per grandi dataset la quantità di memoria richiesta può essere enorme. La complessità di questa classe di metodi e le sue prestazioni dipendono dal numero di vicini più vicini considerati.

3.4 Regressione localmente ponderata (locally weighted regression)

Anche i metodi *locally weighted regression* fanno parte della famiglia degli algoritmi lazy learning. Come per i metodi basati sulle istanze, la previsione viene eseguita utilizzando un sottoinsieme delle istanze nel training set. Quindi, le istanze del training set, che sono rappresentate come punti nello spazio Euclideo D-dimensionale, influenzano

fortemente la previsione su base locale. La differenza principale tra i metodi instance-based e i metodi locally weigthed è che, mentre i primi calcolano la previsione tramite la media degli elementi del training set più vicini alla posizione considerata, i metodi locally weighted eseguono la predizione utilizzando un modello stimato localmente. I modelli locali sono generalmente funzioni parametriche lineari o non lineari. Questi metodi si basano anche sulla ponderazione dei dati per dare più importanza agli esempi rilevanti e meno importanza agli esempi meno rilevanti. Lo stesso effetto può essere ottenuto anche replicando le istanze importanti. La rilevanza è calcolata, analogamente alla somiglianza dei metodi basati sulle istanze, misurando la distanza tra una nuova istanza e ciascun punto del training set. Le funzioni utilizzate per pesare il contributo dei punti del training sono chiamati *kernel*, uno delle più utilizzate è la funzione gaussiana.

I metodi locally weighted hanno dimostrato di essere maggiormente flessibili e con interessanti caratteristiche, come la morbidezza, rispetto ai metodi basati su istanza. Tuttavia, la scelta di un'appropriata funzione di similarità è critica per le prestazioni del modello. Questo può essere un problema in quanto è difficile formalizzare quando due punti sono da considerarsi vicini. La complessità computazionale della fase di training è ridotta al minimo (si tratta solo di memorizzare l'insieme dei dati), ma la fase di previsione può essere computazionalmente costosa. Per limitare il costo della previsione per grandi insiemi di dati, possono essere utilizzate delle strutture di dati idonee alla memorizzazione del training set. Ad esempio, potrebbe essere utilizzata come struttura il *kd-tree*. Un kd-tree è una struttura dati di partizionamento dello spazio, utile a organizzare i punti in uno spazio k-dimensionale. Si tratta di un albero binario dove ogni nodo rappresenta un punto k-dimensionale. I nodi non foglia possono essere visti come una suddivisione del piano in semipiani. I punti che sono a sinistra del semipiano sono quelli nel sottoalbero sinistro del nodo, quelli che sono a destra sono quelli del sottoalbero destro. Viene utilizzato soprattutto per ridurre il tempo di ricerca dei punti di rilevanza per un'istanza.

3.5 Reti Neurali Artificiali

Le reti neurali artificiali (ANN - *Artificial Neural Networks*) sono una famiglia di modelli appartenenti alla classe degli approcci parametrici, e s'ispirano al modello computazionale della rete neurale del cervello umano. Tra i modelli di questa famiglia, il più comune è il *feedforward multilayer perceptron* (MLP). Questo modello è composto da una sequenza di strati di unità computazionali, chiamati neuroni, che svolgono una

semplice elaborazione degli input che ricevono. Ciascun neurone è caratterizzato da una funzione, chiamata funzione di attivazione, che viene utilizzata per il calcolo della sua produzione. Per questo scopo vengono comunemente usate le funzioni lineari e logistiche:

$$B(z) = \frac{1}{1+e^{-z}}$$

In figura 3.1 è riportato uno schema del modello MLP. Il MLP elabora i dati di un opportuno strato: ogni strato riceve in input l'output dello strato precedente e fornisce l'input allo strato successivo. Solitamente, sono collegati solo i neuroni degli strati consecutivi. Ogni connessione è caratterizzata da un parametro scalare chiamato peso, che è un coefficiente moltiplicativo che viene applicato al valore trasmesso dal collegamento stesso. L'ingresso di ogni unità è la somma ponderata di tutti i contributi portati dalle connessioni in ingresso, di conseguenza è una combinazione lineare degli output delle unità dello strato precedente. Il primo e l'ultimo strato sono generalmente composti da neuroni lineari, mentre gli strati intermedi, detti strati nascosti, sono composti da neuroni non lineari.

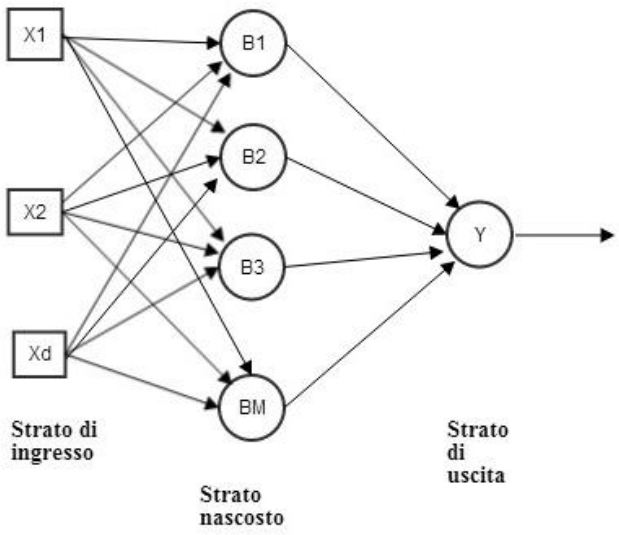


Figura 3.1 – *Schema di un modello di rete neurale feedforward con un singolo strato nascosto.*

I modelli MLP possono essere utilizzati per risolvere efficientemente una vasta classe di problemi di regressione e classificazione. È stato dimostrato che il MLP con almeno uno strato nascosto è un approssimatore universale. Tuttavia l'aggiunta di strati nascosti in più può consentire di eseguire un'approssimazione più efficiente con meno unità. L'uscita della rete viene calcolata come:

$$\hat{f}(x) = \sum_{m=1}^{M} \beta_m B\left(\gamma_m^T \cdot x\right)$$

dove M è il numero di unità nascoste, β_m è il peso (uno scalare) della connessione tra l'unità di uscita e la m-esima unità nascosta, γ_m è il vettore dei pesi delle connessioni tra le unità di ingresso e la m-esima unità nascosta.

In generale, le reti neurali possono avere schemi più complessi di quello mostrato nella figura. Per esempio, nelle reti neurali più ricorrenti l'uscita è retroazionata all'ingresso o agli strati intermedi. Nelle reti completamente connesse, l'ingresso delle unità negli strati inferiori è inviato a tutte le unità degli strati superiori e non solo a quelle di quello successivo. Il numero di unità nascoste di una rete determina il trade-off tra distorsione e varianza del modello. Più sono le unità nascoste utilizzate in una rete più i dati di training risultano adatti. Tuttavia, se viene utilizzato un numero troppo grande di unità, il modello può mostrare overfitting e una scarsa capacità predittiva.

I parametri di una rete neurale vengono trovati generalmente minimizzando una *loss function* che misura la distanza tra l'uscita misurata nell'insieme dei dati d'ingresso e l'uscita calcolata dalla rete per gli stessi valori di ingresso. La funzione di distanza solitamente adottata è la somma dei quadrati (distanza euclidea).

Poiché l'uscita del MLP non è lineare (e la loss function corrispondente non è quadratica) rispetto ai parametri del modello, il valore ottimale dei parametri non può essere trovato direttamente (ad esempio, come soluzione di un sistema lineare). Viene di solito eseguita un'ottimizzazione iterativa, basata sul calcolo del gradiente della loss function. Il gradiente può essere calcolato in vari modi, l'algoritmo più popolare è algoritmo di *backpropagation* a causa della sua efficienza computazionale. La determinazione dei parametri attraverso procedure iterative di ottimizzazione è chiamata *learning*, nella terminologia delle reti neurali. Altri termini equivalenti spesso utilizzati in questo campo sono *configuration*, che viene utilizzato anche quando nella procedura di ottimizzazione viene considerato il numero di unità, e *training*, che mette l'accento sull'uso iterativo dei dati per il calcolo dei parametri cambiando il loro valore con quello ottimale.

3.6 Regressione tramite l'induzione su regole (rule induction regression)

I metodi chiamati *rule induction regression* sono stati sviluppati per quei problemi di regressione che forniscono delle soluzioni interpretabili. Questi metodi hanno lo scopo di estrarre le regole da un dato insieme di training. Un formato comune per le soluzioni interpretabili è il modello *Disjunctive Normal Form* (DNF). I modelli di rule induction sono stati inizialmente introdotti per i problemi di classificazione e successivamente estesi ai problemi di regressione.

Questi approcci hanno delle caratteristiche molto simili agli alberi di decisione, in quanto entrambi i modelli inducono una suddivisione ricorsiva dello spazio d'ingresso, ma gli algoritmi di rule induction, generalmente, forniscono modelli con migliori capacità esplicative poiché le regole presenti non si escludono a vicenda. Queste regole sono più compatte ed hanno un maggiore potere predittivo rispetto agli alberi di decisione, ma questi ultimi mostrano migliori performance quando esistono delle forti dipendenze tra variabili d'ingresso. Gli alberi di decisione sono molto adatti a trovare le variabili di input rilevanti nei casi *high dimensional* quando soltanto un piccolo sottoinsieme di esse è significativa; uno dei loro svantaggi è che il partizionamento dello spazio d'ingresso può causare discontinuità nella predizione nelle regioni di confine.

La soluzione fornita da un albero di regressione è generalmente composta da un pool di regole *if-then*, come, ad esempio:

$$\text{IF } x_1 \in R_k \text{ THEN } \hat{f}(x) = a_k = median\left\{y_i^k\right\}$$

dove R_k è una regione dello spazio degli input, a_k è una costante, e $\left\{y_i^k\right\}$ è l'insieme dei punti del training set che si trovano all'interno di R_k. In questo esempio, il valore della variabile di uscita per il campione x viene calcolato come media delle variabili di uscita dei punti del training set che si trovano all'interno di R_k. Questa è una scelta comune in quanto la mediana minimizza la distanza media assoluta.

La procedura generale per ottenere l'albero di regressione è descritta nel seguito. L'albero viene inizializzato associando l'intero training set alla radice. Poi una procedura ricorsiva di splitting (divisione) viene applicata a ciascun nodo, finché la cardinalità dell'insieme di dati associati a ciascun nodo è sotto una certa soglia. A questo scopo, per ciascun nodo, viene applicata la miglior suddivisione unica (per esempio, quella che minimizza la distanza media assoluta degli esempi del training del sottoinsieme

partizionato), generando due figli per ogni nodo. Poiché l'obiettivo è di trovare l'albero che meglio generalizza nuovi casi, si esegue una seconda fase di potatura (fase di pruning) per eliminare i nodi che causano overfitting.

Nel campo degli alberi di regressione ogni singola partizione di R_k rappresenta una regola e l'insieme di tutte le partizioni disgiunte è denominato *rule-set*. Nell'approccio rule induction, invece, le partizioni per le regole non devono essere separate: un singolo campione può soddisfare diverse regole. Quindi è necessaria una strategia per scegliere una sola regola da applicare tra quelle che sono soddisfatte. Le regole possono essere ordinate secondo un dato criterio (ad esempio, l'ordine di creazione). Tale rule-set ordinato si chiama *decision list*. Successivamente, viene selezionata la prima regola nell'elenco.

Il modello di regressione rule induction può essere costruito, come l'albero di regressione, aggiungendo un nuovo elemento alla volta (cioè quello che minimizza la distanza). Ogni regola viene estesa fino a quando il numero di esempi del training set, che sono coperti dalla regola, non scende al di sotto di una determinata soglia. I casi che vengono coperti sono rimossi e continua il processo di induzione sui restati casi.

Viene presentato di seguito un algoritmo di regressione rule induction basato sull'algoritmo di classificazione *Swap-1*. Swap-1 inizia con una scelta casuale dell'insieme di variabili di input per creare una regola candidata e scambia tutte le componenti congiunte con tutte le possibili componenti. Tale scambio comprende la cancellazione di alcuni componenti dalla regola candidata. La ricerca termina quando non ci sono più possibili scambi che migliorano la regola candidata, tenendo conto che la valutazione di ogni regola candidata si basa sulla capacità di prevedere il valore delle variabili di uscita sugli esempi che soddisfano la condizione della regola. Ogni volta che una nuova regola viene aggiunta al modello, tutti gli esempi che la soddisfano vengono rimossi. Questo procedimento viene iterato fino a quando il training set diventa vuoto. Viene eseguito un pruning dopo la creazione del rule-set: se la cancellazione di una regola non diminuisce la precisione del modello, allora la regola viene eliminata.

Dato che lo Swap-1 è stato progettato per le variabili di input nominali, è necessaria una pre-elaborazione per mappare le variabili di input numeriche in quelle nominali. Questo è realizzato utilizzando una variante dell'algoritmo *k-means*: vengono ordinati i valori delle uscite dei punti y_i. Successivamente viene assegnato ad ogni classe un egual numero di valori contigui di y_i. Un punto passa nella classe adiacente se si riduce la distanza tra il punto e la media dei punti della classe.

L’algoritmo di regressione rule induction viene realizzato utilizzando l’algoritmo Swap-1 sui dati mappati e successivamente si occupa di effettuare il pruning e l’ottimizzazione del rule-set. Dopo il pruning, viene eseguita l'ottimizzazione ricercando il miglior sostituto per ogni regola in modo tale che l'errore di previsione si riduca. Il valore di predizione può essere calcolato come la mediana o il valore medio degli elementi della classe, ma può essere applicato un modello parametrico o non parametrico a ciascuna classe.

Nel prossimo capitolo verrà introdotta una tipologia di sistemi a regole che utilizza un particolare tipo di regole, le *regole fuzzy*. Il sistema di questo tipo utilizzato in questo lavoro di tesi è il sistema di Takagi-Sugeno.

4 SISTEMI A REGOLE FUZZY

La logica Fuzzy o logica sfumata è una logica in cui si può attribuire a ciascuna proposizione un grado di verità compreso tra 0 e 1. Con grado di verità o valore di appartenenza si intende quanto è vera una proprietà: questa può essere, oltre che vera (valore 1) o falsa (valore 0), come nella logica classica, anche pari a valori intermedi. Il termine "fuzzy logic" è stato introdotto nel 1965 in una pubblicazione sulla teoria degli insiemi fuzzy da Lotfi A. Zadeh, professore all'Università della California di Berkeley. Nella teoria classica degli insiemi la composizione degli elementi di un insieme è valutata in termini binari, cioè un elemento o appartiene o non appartiene a un insieme. La teoria degli insiemi fuzzy invece permette una valutazione graduale dell'appartenenza o meno di un elemento a un insieme e questo è possibile con la definizione di una funzione di appartenenza μ valutata nell'intervallo [0, 1]

$$A = \{(x, \mu_A(x)) \mid x \in X\}$$

dove X è l'universo di definizione, che è un insieme convenzionale (o crisp) e A un insieme fuzzy.

Possono essere usate diverse funzioni per rappresentare le funzioni di appartenenza, come triangoli, trapezi, sigmoide, funzione di distribuzione gaussiana.

Le operazioni logiche tra insiemi sono ridefinite, in misura tale da essere valide anche con i classici insiemi crisp:

- A and B → min(A,B) o prod(A,B)
- A or B → max(A,B) o sum(A,B)
- not A → 1 - A

Mentre le variabili in matematica solitamente assumono valori numerici, nelle applicazioni di logica fuzzy sono spesso utilizzate le variabili linguistiche (non-numeriche) per facilitare l'espressione di regole e fatti.

Una variabile linguistica è caratterizzata da una quintupla

(v, T, X, g, m)

dove *v* = nome della variabile

T = insieme dei termini linguistici di *v*;

X = universo di definizione della variabile base;

g = regola sintattica (una grammatica) per generare i termini linguistici (ad esempio, "molto alta", "non molto bassa");

m = regola semantica che assegna a ciascun termine linguistico $t \in T$ il suo significato, *m(t),* che è un insieme fuzzy su X cioè $m : T \to F(X)$.

Una variabile linguistica è definita attraverso i suoi termini primari che sono etichette di insiemi fuzzy (ad esempio "bassissima", "bassa", "media", "alta" e "altissima").

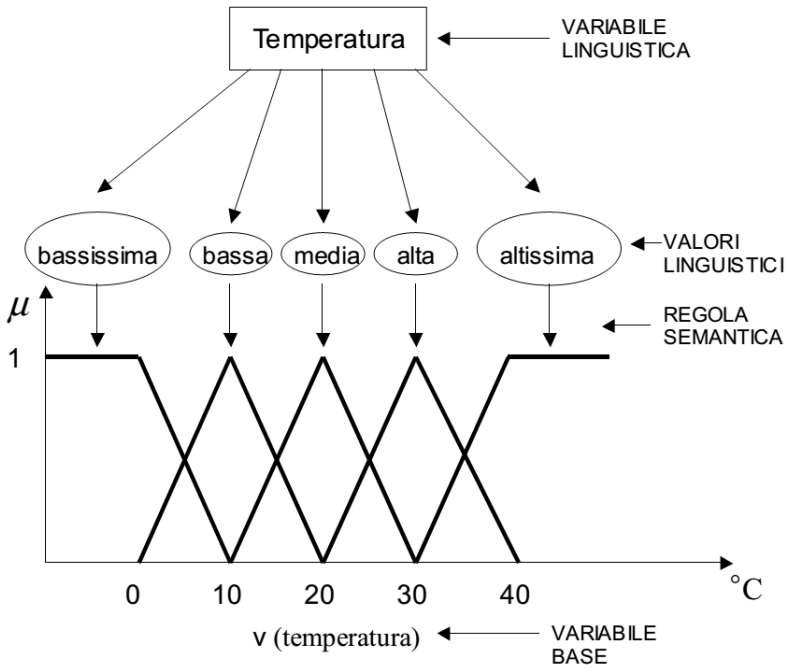


Figura 4.1 – *Esempio di variabile linguistica.*

La logica fuzzy di solito usa delle regole dette regole IF-THEN. Queste regole sono generalmente espresse nella forma:

IF *x* ***IS*** *A* ***THEN*** *y* ***IS*** *B*

dove A e B sono espressioni linguistiche che identificano degli insiemi fuzzy sui domini di x e y. La prima parte della regola "*x IS A"* è detto *antecedente* mentre la seconda parte, *"y IS B"* è detto *conseguente*. Se l'antecedente è vero con un certo grado di appartenenza, allora il conseguente è altrettanto vero con lo stesso grado di appartenenza. Nel caso in cui l'antecedente è costituito da più parti, quest'ultime devono essere associate ad un singolo numero attraverso l'uso degli operatori fuzzy. Se invece è il conseguente ad

essere costituito da più parti allora tutti i conseguenti sono influenzati equamente dal risultato dell'antecedente.

Si definisce inferenza fuzzy il processo di mappatura da uno spazio di input dato a un opportuno spazio di output utilizzando la logica fuzzy.

4.1 Il processo di inferenza fuzzy

Il processo di inferenza fuzzy si può dividere in cinque passi:

1. Fuzzyficazione delle variabili di input: consiste nella valutazione di una funzione o nella consultazione di una tabella;
2. Applicazione degli operatori fuzzy (AND oppure OR) nell'antecedente: se l'antecedente di una regola ha più di una parte, si applicano uno o più operatori fuzzy per ottenere un unico numero che rappresenti il risultato dell'antecedente per quella regola. L'input per l'operatore fuzzy è costituito da due o più valori di appartenenza delle variabili di input fuzzyficate e l'output è costituito da un singolo valore di verità;
3. Applicazione del metodo d'implicazione dall'antecedente al conseguente: l'input per il processo di implicazione è un numero dato dall'antecedente, e l'output è un insieme fuzzy; tale insieme si ottiene combinando l'insieme fuzzy indicato nel conseguente con il valore dell'antecedente attraverso gli operatori min o prod;
4. Aggregazione dei conseguenti delle regole: l'aggregazione è il processo mediante il quale gli insiemi fuzzy che rappresentano gli output di ogni regola sono combinati per dare luogo ad un singolo insieme fuzzy. Gode della proprietà commutativa ed è implementato attraverso le funzioni max oppure sum;
5. Defuzzyficazione: in tale fase si ricava il valore di output numerico dall'insieme fuzzy di output aggregato.

I principali sistemi di inferenza più usati sono Mamdani e Takagi-Sugeno. Nel primo il conseguente della regola è costituito anch'esso da un insieme fuzzy, mentre nel secondo il conseguente è costituito da una combinazione lineare degli ingressi.

In questo lavoro di tesi verrà utilizzato il sistema di Takagi-Sugeno (TS).

4.2 Approccio Takagi-Sugeno

I sistemi Takagi-Sugeno, originariamente proposti da Takagi e Sugeno nel 1985, sono costituiti da una base di regole fuzzy IF-THEN che suddividono lo spazio in regioni fuzzy descritte dagli antecedenti. Il conseguente di ogni regola *i* è un'espressione del modello degli ingressi. Il modello TS può essere visto come una mappatura che va dallo spazio dell'antecedente a una regione convessa nello spazio del conseguente, definito dalle variazioni dei parametri del conseguente. L'i-esima regola fuzzy ha la seguente forma:

$$\mathrm{R}^{(\mathrm{i})}:\ \mathrm{IF}\, x_1\, is\, A_1\, AND \ldots AND\, x_n\, is\, A_n \mathrm{THEN}\, y_i = f_i(x)$$

dove $f_i(x)$ è una combinazione lineare degli ingressi (es. ax_1+bx_2+c). In questo elaborato si utilizza il TS di ordine zero in cui l'output del sistema è una costante ($a=b=0$).

Il livello dell'output y_i di ogni regola è ponderato dalla forza w_i della regola. Ad esempio per una regola AND con input x_1 e x_2, il peso della stessa è:

$$w_i = AndMethod\left(A_1(x_1), A_2(x_2)\right)$$

dove A_1 ed A_2 sono le funzioni di appartenenza di x_1 e x_2.

L'output finale del sistema è la media pesata degli output delle regole calcolata come:

$$Final\ Output = \frac{\sum_{i=1}^{N} w_i y_i}{\sum_{i=1}^{N} w_i}$$

dove N è il numero di regole.

Nelle seguenti figure viene riportato il flusso di operazioni (flow chart) seguito in questo lavoro di tesi per la visualizzazione di curve, nel caso 2D, e superfici, nel caso 3D, da una nuvola di punti.

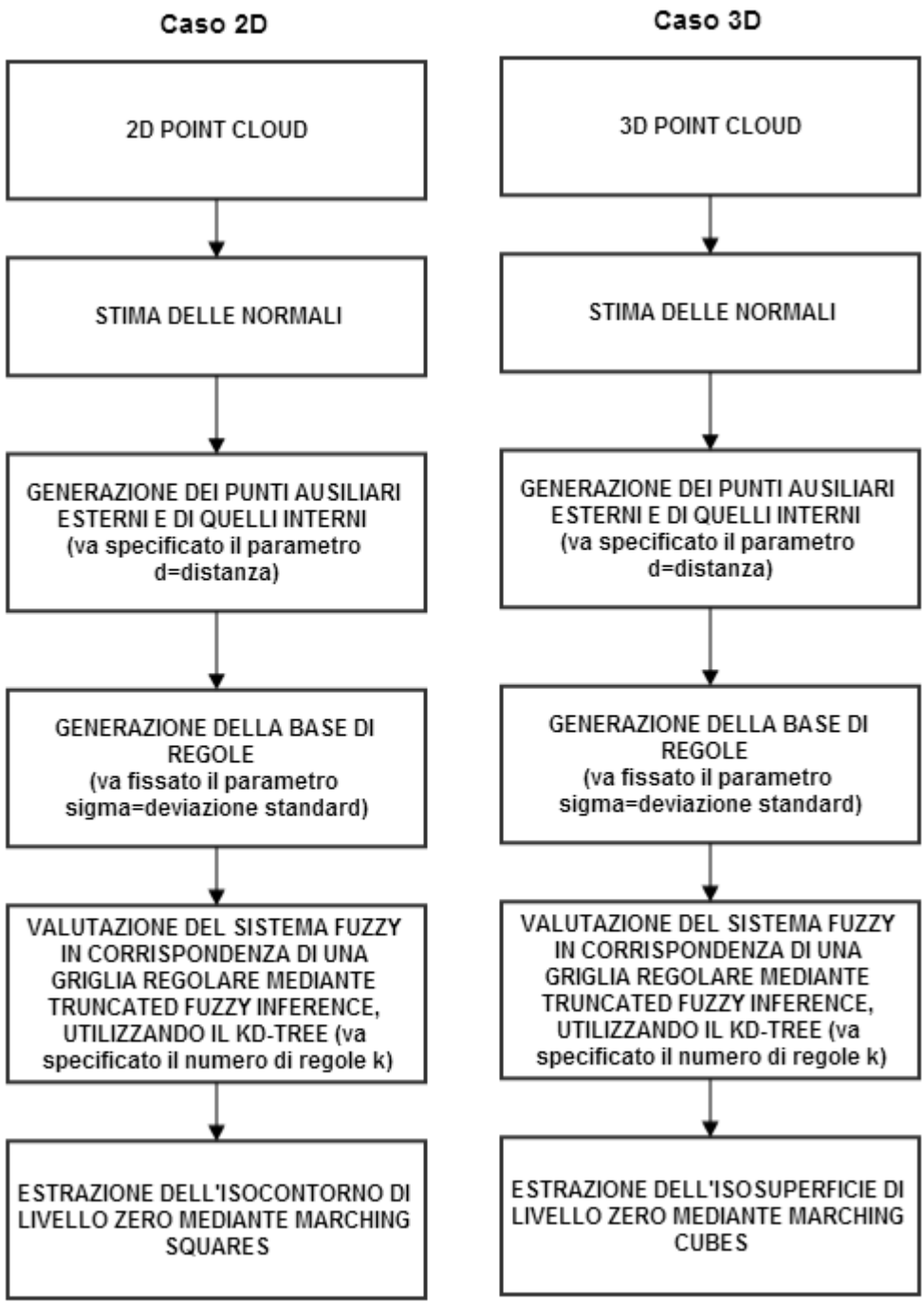


Figura 4.2 – *Flow chart per il caso in R^2e per il caso in R^3*

5 UN METODO VELOCE ED EFFICIENTE PER L'INFERENZA FUZZY

In questo elaborato vengono utilizzati i sistemi TS per la ricostruzione di oggetti tridimensionali acquisiti tramite scanner 3D.

L'approccio seguito è basato sulla costruzione di un modello analitico della funzione, data in maniera implicita. E' quindi un approccio basato su modellizzazione, poiché si cerca di creare un modello implicito e globale della funzione che si vuole approssimare. Con la costruzione del modello implicito e globale è più facile approssimare la figura in zone in cui i punti non sono presenti. Nel caso del 3D non abbiamo la funzione ma una nuvola di punti ricavati tramite scannerizzazione in 3D di un oggetto. Quando scannerizziamo dei punti, ci troviamo nella situazione in cui non abbiamo un campionario uniforme. Per permettere una buona visualizzazione sarà necessario aggiungere ai vettori x_i y_i e z_i altri punti che saranno positivi se esterni alla superficie e negativi se interni, in modo da avere maggiori informazioni soprattutto nei casi di mancanza di dati. La prima parte dell'algoritmo è incentrata sulla creazione di questi punti ausiliari, sulla scelta della distanza di questi punti dalla superficie e sulla creazione di una griglia regolare. Successivamente con l'utilizzo dei sistemi TS di ordine 0 si crea il modello e di conseguenza le regole fuzzy. Si procede poi con la valutazione delle regole su tutta la griglia.

Si è scelto di usare varie funzioni presenti in MATLAB e altre funzioni già sviluppate, come ad esempio il *K-NN search,* il *KdTree* e il *Marching Cubes* (o *Marching Squares* nel caso del 2D). E' stata introdotta per la prima volta la *Truncated Fuzzy Inference* (TFI), l'inferenza fuzzy con troncamento. Di seguito verrà descritta più in dettaglio ognuna di queste funzioni e come sono state utilizzate all'interno dell'algoritmo proposto.

5.1 Passi dell'algoritmo

L'algoritmo è composto da tre passi. Nel primo passo viene caricata la nuvola di punti da un file, un dataset contenente le coordinate dei punti e le relative coordinate delle

normali ai punti. Viene poi effettuato il *SubSampling* dei dati che consiste nella riduzione dei punti che compongono l'immagine.

Nel passo successivo viene creata la base di regole fuzzy per passare poi alla valutazione su una griglia regolare. Questo viene fatto dalla funzione chiamata "*FitS2D"* o *"FitS3D"*, rispettivamente nel caso bidimensionale e tridimensionale.

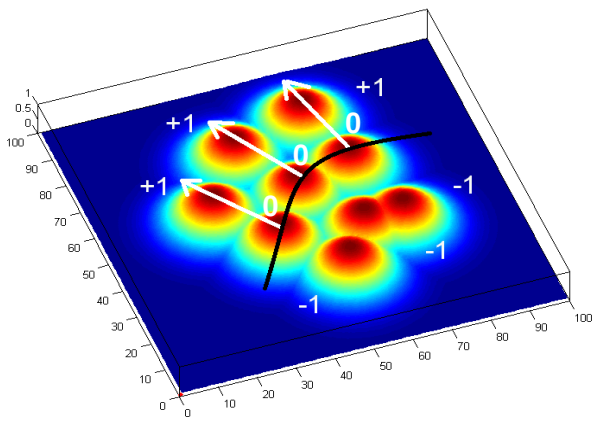
Nel terzo e ultimo passo dell'algoritmo viene fatto il *rendering* dell'immagine ottenuta con l'utilizzo del *Marching Squares* o *Marching Cubes* di Matlab.

5.2 Funzione fitS2D o fitS3D

Entrambe queste funzioni si occupano di ricostruire l'oggetto dai punti; la fitS2D si occupa della ricostruzione della curva mentre la fitS3D della superficie. Da ora in avanti si parlerà in particolare della fitS3D, in quanto entrambe le funzioni svolgono i medesimi passi con la differenza che nella fitS2D si utilizzano solo le coordinate *x* e *y* mentre nella fitS3D si aggiunge la terza coordinata *z*.

Inizialmente viene calcolata la Bounding box e successivamente tramite la funzione *meshgrid* viene creata la griglia 3D regolare.

Per avere una miglior visualizzazione si è scelto di creare dei punti ausiliari, e più in particolare dei punti PAp esterni alla superficie e dei punti PAm interni. Scelta una distanza *d*, si calcolano, per ogni punto del dataset, due punti che si trovano a questa distanza lungo la normale del punto considerato, uno nel verso uscente e uno nel verso entrante, creando così i vettori di punti ausiliari da aggiungere al training set per l'addestramento (Fig. 5.1).



5.1 – *Creazione dei punti ausiliari e rispettive gaussiane. Ai punti esterni si assegna valore +1, mentre ai punti interni si assegna valore -1. Le frecce bianche rappresentano le normali alla curva nel punto considerato. Ai punti originali, posti sulla superficie, viene invece assegnato il valore 0.*

Vengono poi creati i punti *target* (Ti), i *target esterni* (TAp) ed *interni* (TAm). Dopodiché si crea il sistema TS di ordine zero (funzione *genTS0*) che prende in ingresso i punti del dataset, i punti ausiliari e tutti i target. Questa funzione restituisce *M* e *PI*, che sono rispettivamente la matrice *mxn* che contiene la media delle gaussiane degli antecedenti della regola (*m* regole con *n* antecedenti per ognuna) e il vettore *mx1* che denota il peso dei conseguenti. Questi verranno passati alla funzione *evalfis_siRBt1ts0_f*, ("Fast evaluation of a zero-order Takagi-Sugeno fuzzy rule-based system, with singleton inputs and type-1 base rule") che di occupa di valutare le regole fuzzy.

5.3 Funzione "evalfis_siRBt1ts0_f"

Questa funzione si occupa della valutazione delle regole fuzzy e in particolare fa uso delle funzioni *knn_search* e *kd-tree*, passando poi alla TFI.

La funzione knnsearch utilizza la ricerca approssimata dei *k* vicini più vicini (ANNSEARCH - Approximate Nearest Neighbor Search); ricerca infatti i punti più vicini ai punti del vettore *X*, passato come argomento, che si trovano nell'insieme dei punti di *M*, non escludendo i punti di *X* che coincidono con quelli di *M*. Per memorizzare il risultato viene creato ad ogni passo dell'algoritmo un kd-tree.

Una volta memorizzate tutte le *k* regole che si attivano per ogni punto della griglia, si passa all'inferenza fuzzy. Occorre innanzi tutto 'fuzzificare' gli input: tipicamente, si considerano i punti come fuzzy singleton.

Tramite la funzione *mygaussmf (x, sigma, c)* vengono create le gaussiane dagli input nella forma:

$$y = e^{-\frac{1}{2}\left(\frac{x-c}{sigma}\right)^2}$$

che vanno a creare la base di regole. Per ogni ingresso viene calcolato il grado di appartenenza a ciascuna regola per ciascuna variabile.

Si usa il prodotto per risolvere l'operatore AND nell'antecedente di ogni regola e infine il massimo come operatore di aggregazione degli insiemi fuzzy prodotti da ciascuna regola.

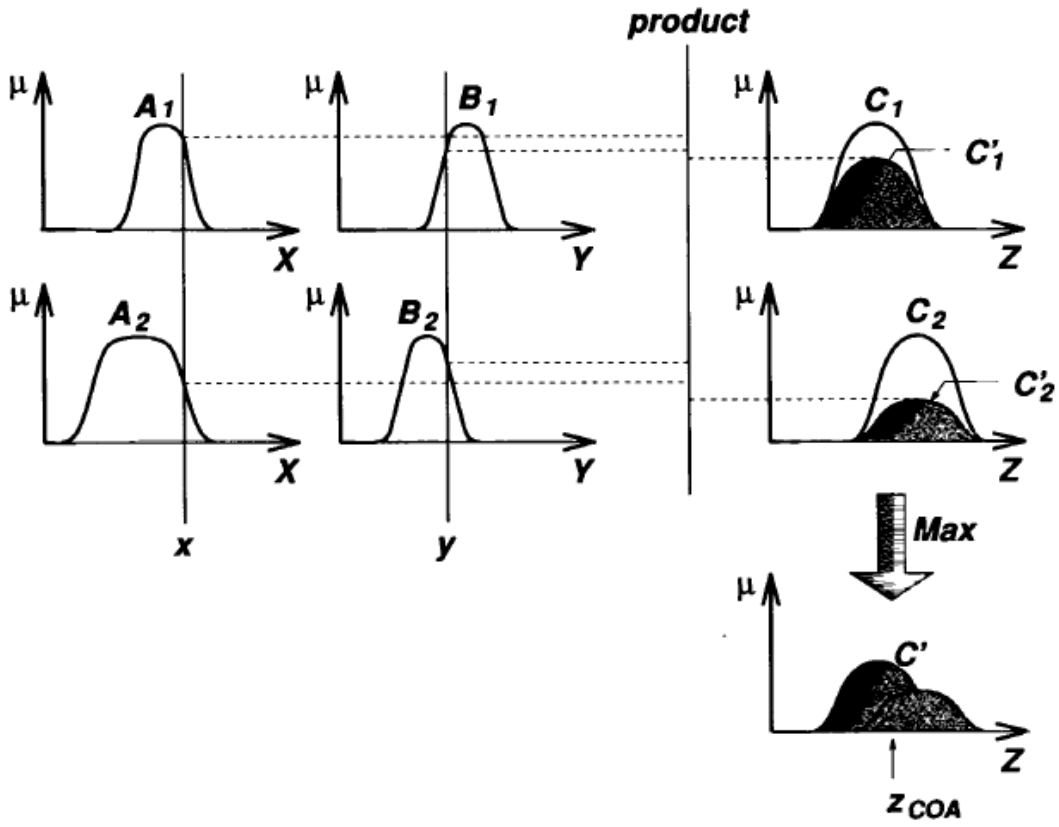


Figura 5.2 – *Esempio di inferenza fuzzy con gaussiane*

Come risultato si hanno così i valori che vengono assegnati tramite inferenza a tutti i punti della griglia, valori che sono vicini allo zero in prossimità della superficie.

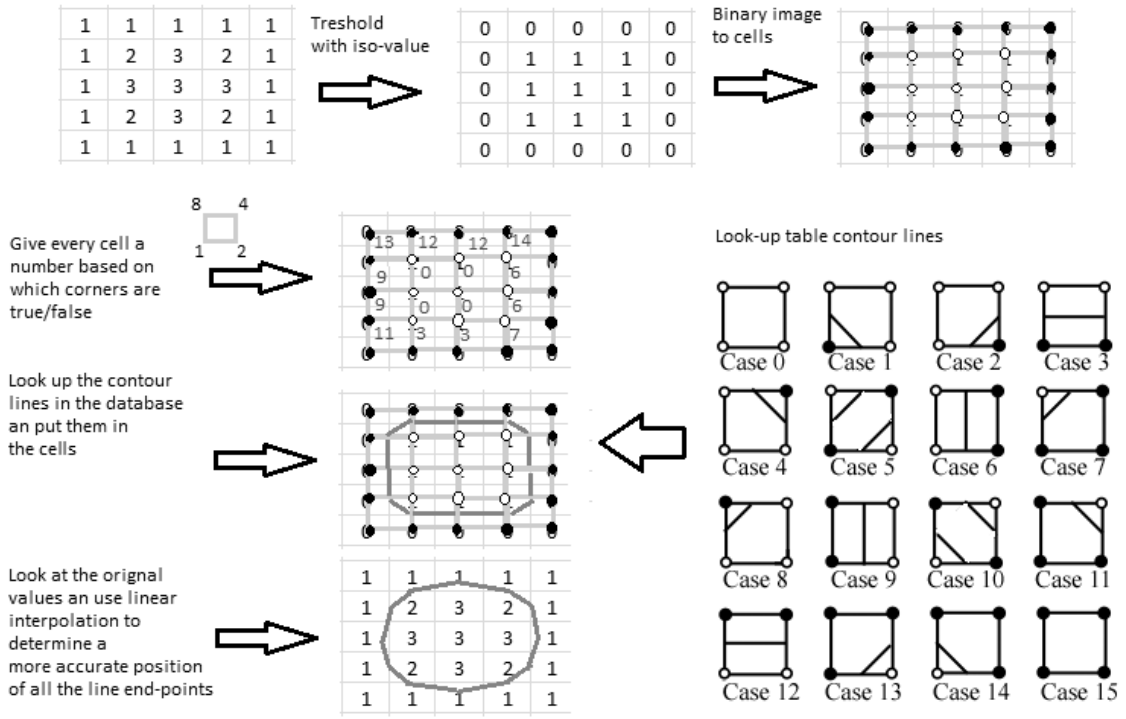
In questo modo è possibile poi utilizzare la tecnica del Marching Cubes o Marching Squares implementato in Matlab per visualizzare l'isosuperficie.

5.4 Marching Cubes e Marching Squares

Questi due algoritmi sono uguali a parte il numero di dimensioni coinvolte, e cioè due nel Marching Squares e tre nel Marching Cubes. In generale l'algoritmo procede attraverso il campo scalare, prendendo quattro/otto locazioni vicine per volta (formando così un quadrato/cubo immaginario), determinando quindi il poligono o i poligoni necessari per rappresentare la parte della isosuperficie che passa attraverso questo quadrato/cubo. I poligoni individuali sono quindi fusi nella superficie desiderata. Questo viene fatto creando un indice in un array precalcolato di 256 configurazioni di poligoni possibili all'interno del cubo, trattando ciascuno degli otto valori scalari come un bit di un

intero di 8-bit. Se il valore dello scalare è più alto dell'iso-valore (cioè è all'interno della superficie) allora il bit appropriato viene posto a uno, mentre se è più basso (esterno) è impostato a zero. Il valore finale dopo che tutti gli otto scalari sono controllati, è l'indice effettivo nella matrice del poligono. Infine ciascun vertice di poligoni generati è messo nella posizione appropriata lungo il vertice del cubo, interpolando linearmente i valori dei due scalari che sono connessi da quel vertice (Figura 5.2).

In Matlab questo viene fatto dalle funzioni isocontour (Marching Squares) e isosurface (Marching Cubes).



5.3 – *Spiegazione grafica del Marching Squares*

6 ESPERIMENTI IN $\Re^2$

In questo capitolo vengono illustrate le simulazioni dell'algoritmo *pcrec2,* implicit curve reconstruction.

6.1 pcrec2

Questa è la funzione che si occupa della ricostruzione di una curva dati i punti e le normali ai punti.

6.1.1 Creazione del dataset: funzione circR1_withNoise

In questo caso il dataset utilizzato viene generato dalla funzione *circR1_withNoise.* Questa funzione crea una nuvola di N punti casuali distribuiti su una circonferenza di raggio unitario e per ogni punto considerato calcola le coordinate polari delle normali a tale punto. Inoltre è possibile inserire come parametro della funzione un valore che permette di introdurre rumore nei punti del dataset. Tutti i punti vengono memorizzati in una matrice 4xN che verrà salvata su un file ".normals". Nella prime due colonne del file ci sono le coordinate *xy* dei punti e nella terza e quarta colonna le rispettive coordinate delle normali.

La funzione è stata eseguita, cambiando i vari parametri, per creare diversi tipi di dataset, in particolare:

- Dataset_1 con 1000 punti e senza rumore ("circ_1000_punti_noise_0.000");
- Dataset_2 con 1000 e con rumore 0.1 ("circ_1000_punti_noise_0.100");
- Dataset_3 con 1000 e con rumore 0.3 ("circ_1000_punti_noise_0.300");
- Dataset_4 con 5 punti e senza rumore ("circ_5_punti_noise_0.000");
- Dataset_5 con 5 punti e con rumore 0.3 ("circ_5_punti_noise_0.300").

In figura abbiamo degli esempi di dataset dove in blu sono rappresentate le normali ai punti.

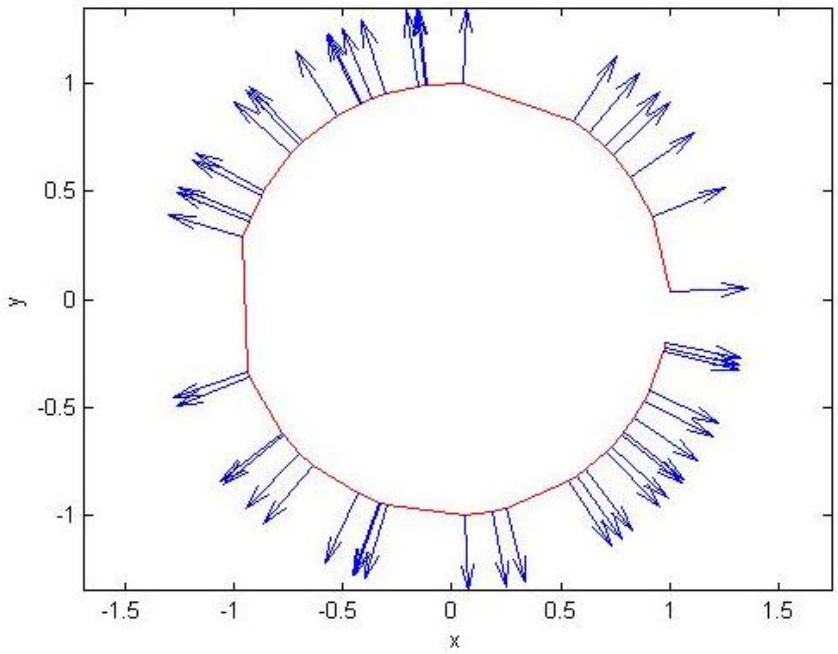


Figura 6.1 - *Esempio dataset con 50 punti e in assenza di rumore*

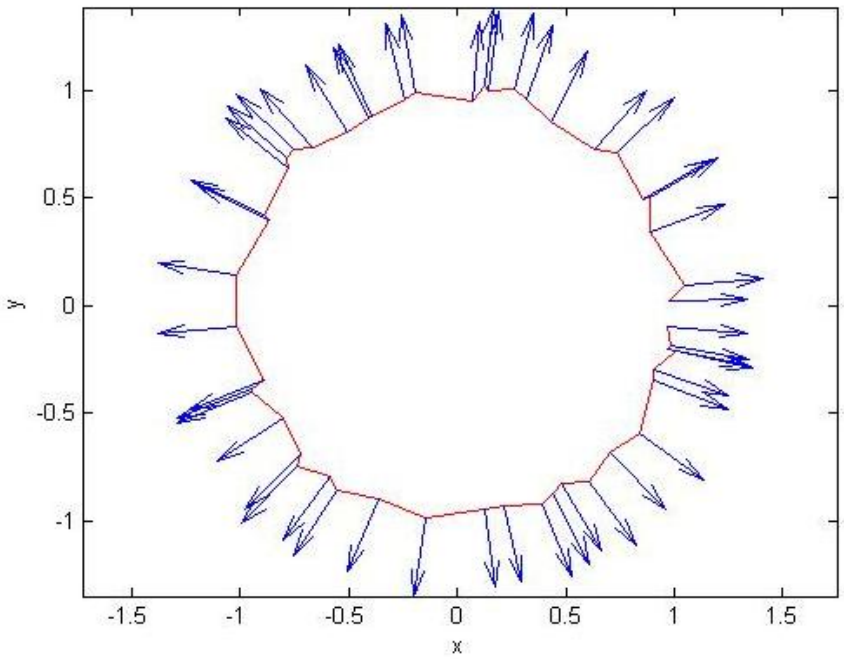


Figura 6.2 - *Esempio dataset con 50 punti e con rumore pari a 0.1*

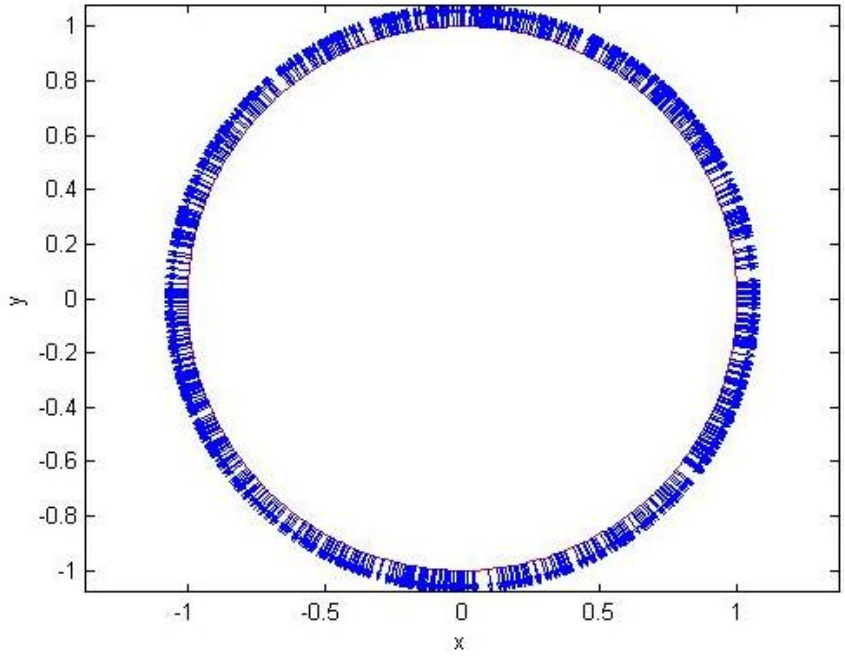


Figura 6.3 – *Esempio dataset con 1000 punti e con assenza di rumore*

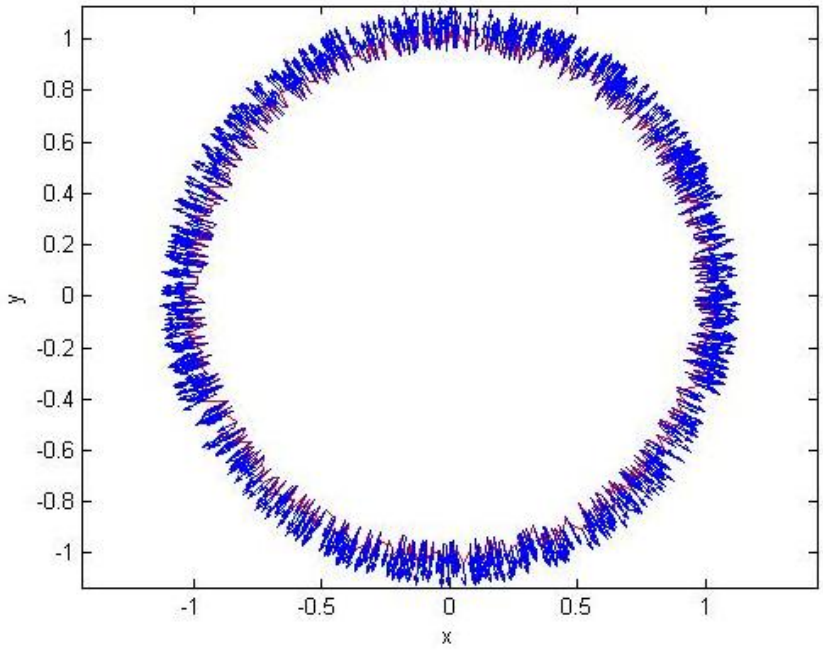


Figura 6.4 – *Esempio dataset con 1000 punti e con rumore pari a 0.1*

6.1.2 Parametri

La funzione *pcrec2* ha i seguenti parametri:

- dataname: nome del file (senza l'estensione ".normals")
- N: grandezza della griglia
- subSamplingFactor: fattore usato per il campionamento dei punti
- k: numero di regole utilizzate per l'inferenza
- sigma: deviazione standard delle funzioni di appartenenza gaussiane
- d: distanza dalla curva dei punti aggiuntivi
- showLine: quando TRUE va in esecuzione il Marching Squares
- profileTheCode: quando TRUE va in esecuzione il profiler
- knnsearchHandler: funzione che college alla funzione k-NN search.
- plotUnitCirc: quando TRUE visualizza la circonferenza unitaria;

Gli esperimenti riportati nei successivi paragrafi sono stati creati tramite un main *(main2D.m)* che manda in esecuzione la funzione del paragrafo 6.1.1 creando i cinque dataset descritti ed esegue la funzione *pcrec2* per questi dataset e in più per un dataset che costituisce uno slice di *bunnyhole.normals*[5].

6.1.3 Calcolo della performance

Nel caso della nuvola di punti su una circonferenza di raggio unitario è stato possibile fare una stima della bontà della ricostruzione. Infatti si conosce la circonferenza ideale ed è quindi possibile calcolare, per ogni punto, la distanza tra la circonferenza ricostruita e quella ideale in questo modo:

$$J_{i+1} = J_i + \sum_{i=1}^{N} \left(\sqrt{(xV_i - xC)^2 + \left(yV_i - yC\right)^2} - rC \right)^2$$

Dove J_{i+1} e J_i sono rispettivamente la differenza attuale e quella dei punti precedenti, xV_i e yV_i sono le coordinate x e y dei vertici dell'i-esimo oggetto (Objects è ottenuto come output della funzione isocontour), *xC, yC* e *rC* sono rispettivamente le coordinate x, y ed il raggio della circonferenza ideale. Questo calcolo viene fatto dalla funzione *circPerf2*.

[5] Questo dataset contiene i punti ottenuti tramite scannerizzazione della statuetta di un coniglio, nel laboratorio di Computer Graphics della Stanford University.

6.1.4 *Dataset 1:* **"circ_1000_punti_noise_0.000"**

Il primo dataset considerato è quello contenente un numero elevato di punti con assenza di rumore.

I parametri impostati sono:

- N = 300;
- SubSamplingFactor = 1;
- K = 200;
- sigma = 70e-3;
- d = 5e-2;
- showLine = true;
- profileTheCode = false;
- knnsearch_handler = @knnsearch_ann_binned;
- plotUnitCirc = true;

Il primo risultato ottenuto è la figura in cui viene visualizzato un esempio di scelta delle regole (Fig. 6.5). L'asterisco blu indica il punto per il quale si stanno valutando le regole, in rosso abbiamo le regole e i quadrati verdi evidenziano le k regole più vicine. Viene poi visualizzato (Fig. 6.6) il risultato ottenuto dalla funzione *surf*, cioè il grafico 3D che rappresenta le curve di livello. Tramite la scala di colori è possibile vedere la curva di livello zero (in nero). Successivamente viene visualizzata la curva tramite *marching squares*. In figura 6.7 abbiamo l'intera circonferenza, mentre in figura 6.8 un particolare per vedere meglio la curva ricostruita dall'algoritmo. In questo caso è possibile vedere anche la differenza tra la circonferenza calcolata dal modello, in nero in figura, e l'effettiva circonferenza di raggio unitario, riportata in magenta.

Tramite la funzione calcPerf2 viene calcolato quanto discosta la circonferenza ricostruita da quella ideale e in questo caso J=0.0099.

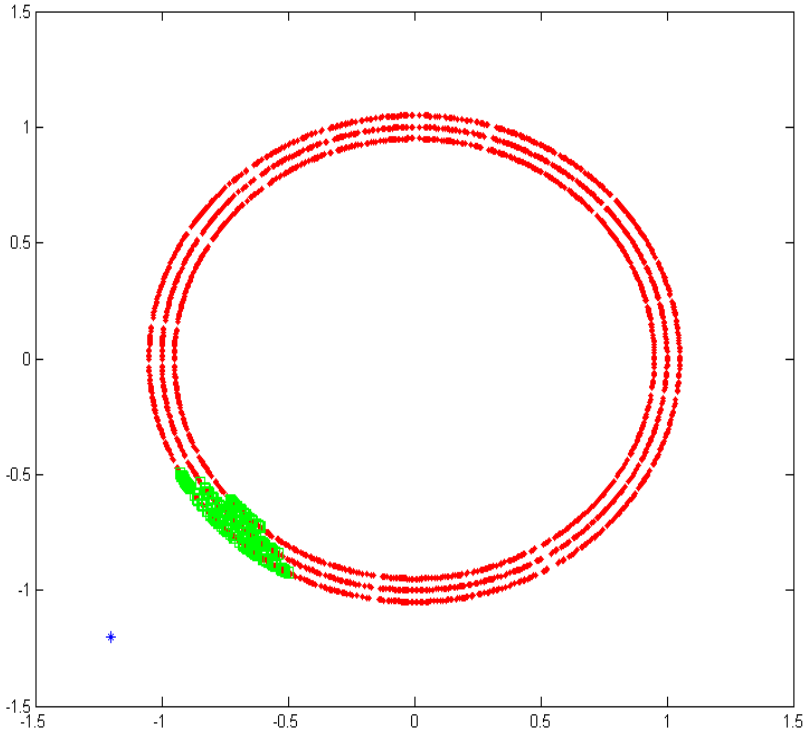


Figura 6.5 – *Selezione delle regole: in rosso si hanno le regole, in blu il punto per il quale si valutano le regole e il quadratino verde evidenzia le regole selezionate per tale punto.*

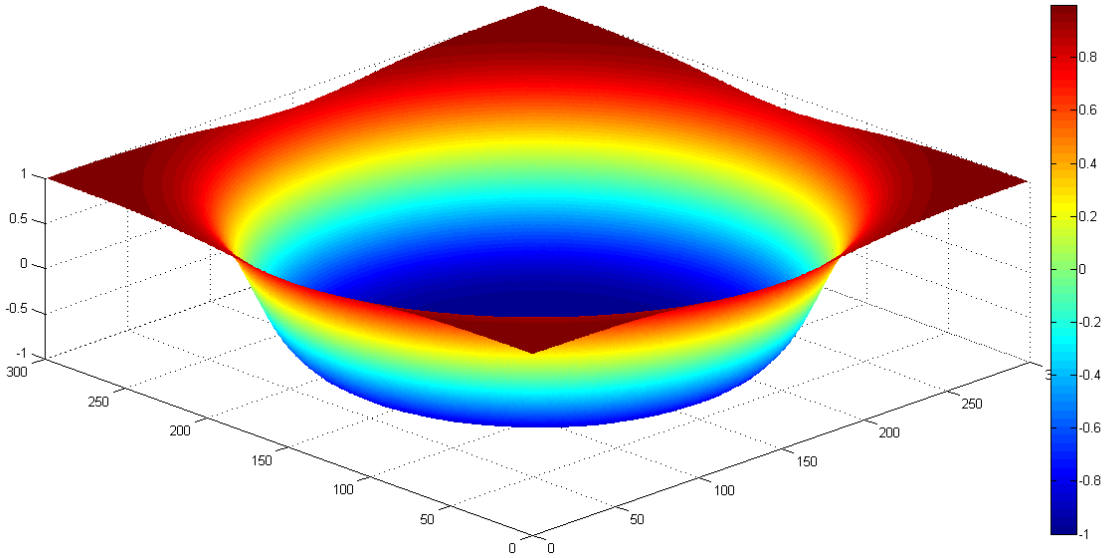


Figura 6.6 – *Grafico 3D delle curve di livello*

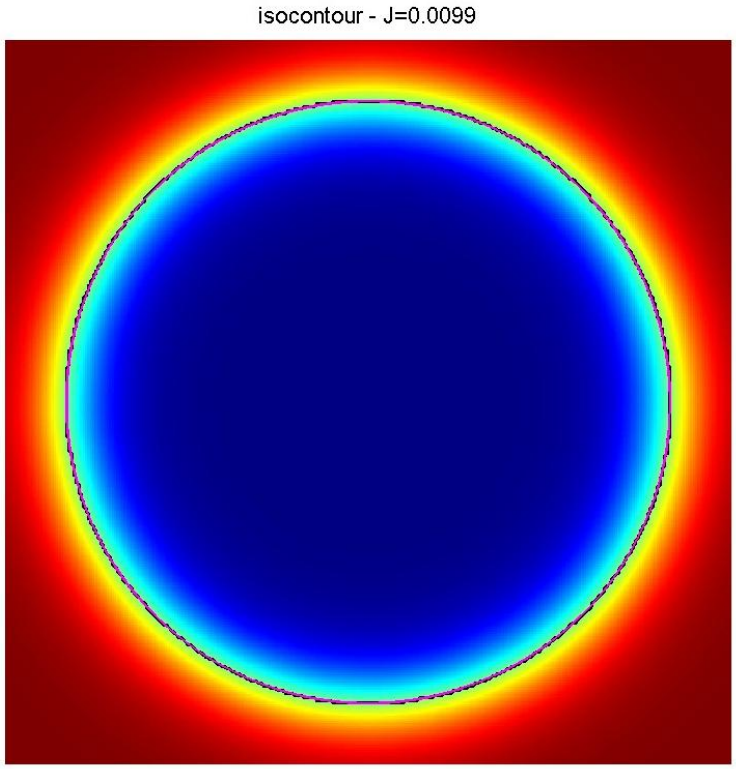


Figura 6.7 – *Visualizzazione del marching squares*

Figura 6.8 – *Particolare della curva. È possibile vedere così la circonferenza disegnata dall'algoritmo (in nero) e la circonferenza di raggio unitario (in magenta).*

6.1.5 *Dataset 2:* “circ_1000_punti_noise_0.100”

Con il secondo dataset si è provato ad introdurre del rumore nei dati, per verificare se anche con dati leggermente rumorosi l’algoritmo riesce a ricostruire abbastanza bene la curva. I parametri sono impostati come per il dataset 1. Viene riportato il grafico 3D delle curve di livello e la ricostruzione della curva tramite l’uso del marching squares. Anche in questo caso (Fig. 6.11) viene effettuato uno zoom per vedere meglio la circonferenza, in nero quella ricostruita tramite l’algoritmo e in magenta la circonferenza di raggio unitario. Possiamo notare che nonostante l’introduzione del rumore, l’algoritmo riesce a ricostruire abbastanza bene la curva, infatti J=0.0453.

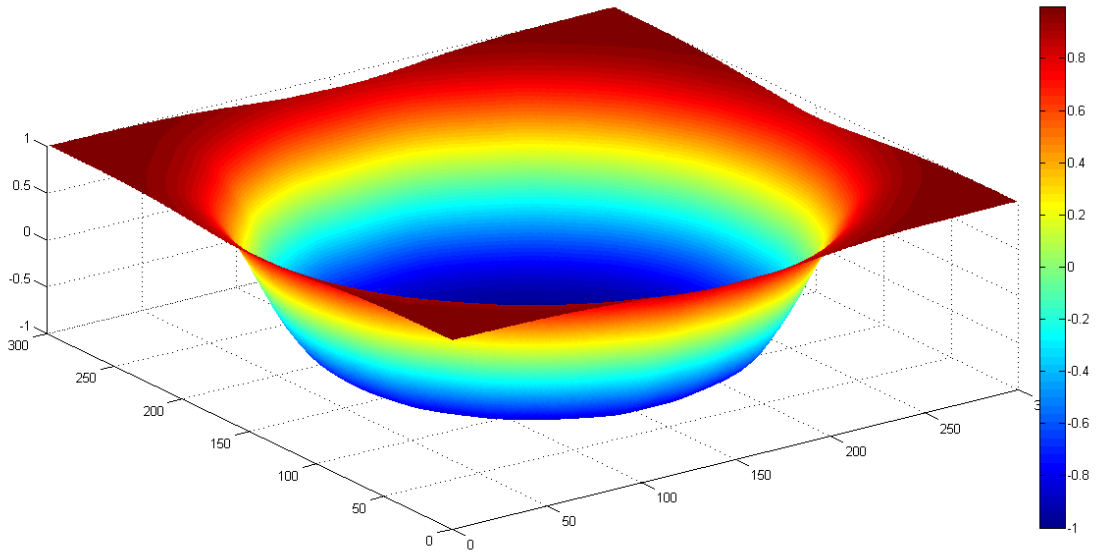


Figura 6.9 - *Grafico 3D delle curve di livello.*

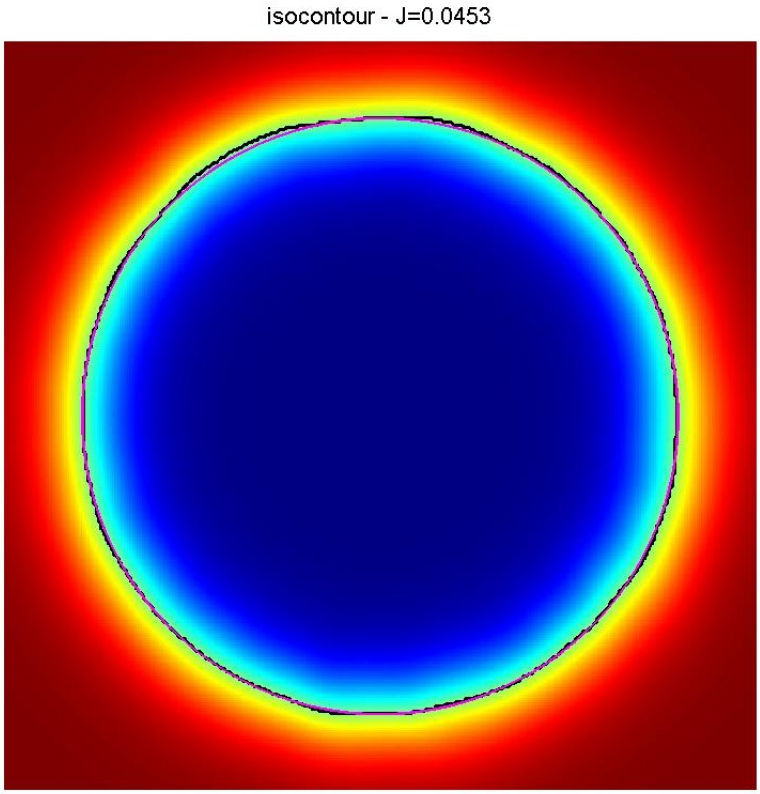


Figura 6.10 - *Visualizzazione del marching squares*

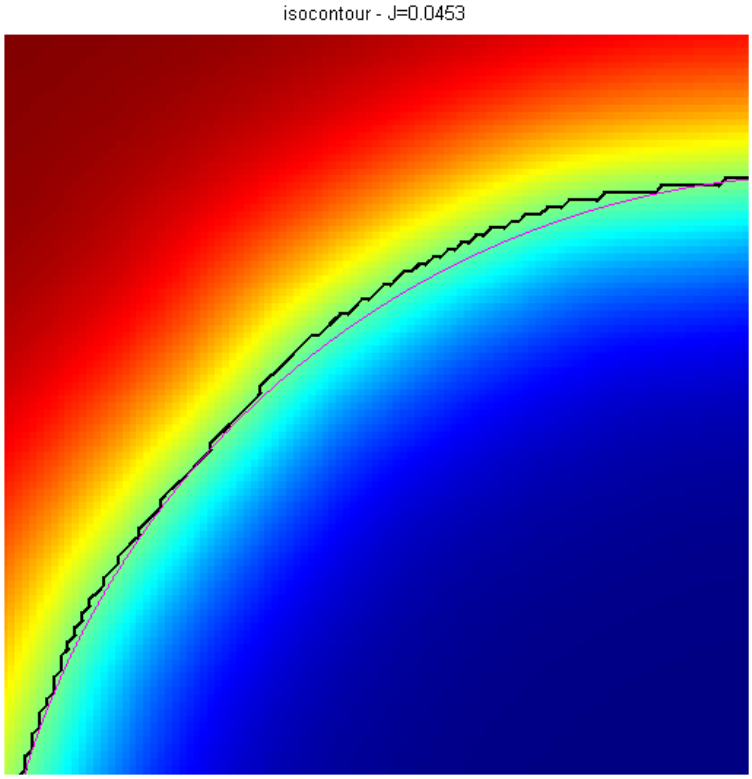


Figura 6.11 - *Particolare della curva. È possibile vedere così la circonferenza disegnata dall'algoritmo (in verde) e la circonferenza di raggio unitario (in magenta).*

6.1.6 Dataset 3: "circ_1000_punti_noise_0.300"

Con il terzo dataset si è provato ad introdurre un rumore ancora più elevato nei dati, per verificare se anche con dati considerevolmente rumorosi l'algoritmo riesce a ricostruire abbastanza bene la curva. I parametri sono impostati come per il dataset 1. Viene riportato il risultato attenuto il grafico 3D delle curve di livello (Fig. 6.12) e la ricostruzione della curva tramite l'uso del marching squares (Fig. 6.13). Anche in questo caso (figura 6.14) viene effettuato uno zoom per vedere meglio la circonferenza, in nero quella ricostruita tramite l'algoritmo e in magenta la circonferenza reale di raggio unitario. Possiamo notare che nonostante l'introduzione di un elevato rumore, l'algoritmo riesce a ricostruire abbastanza bene la curva, infatti J=0.9617

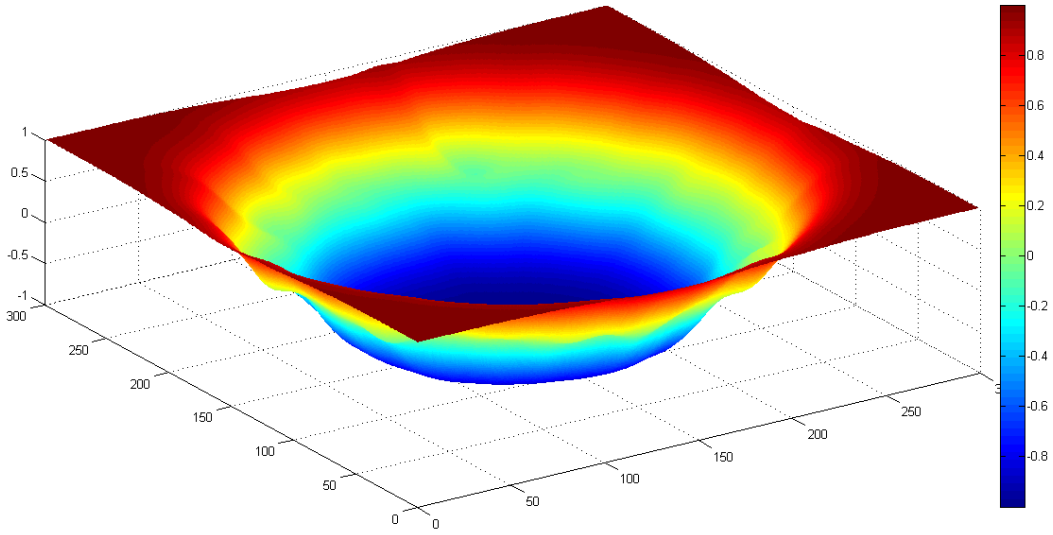


Figura 6.12 - *Grafico 3D delle curve di livello.*

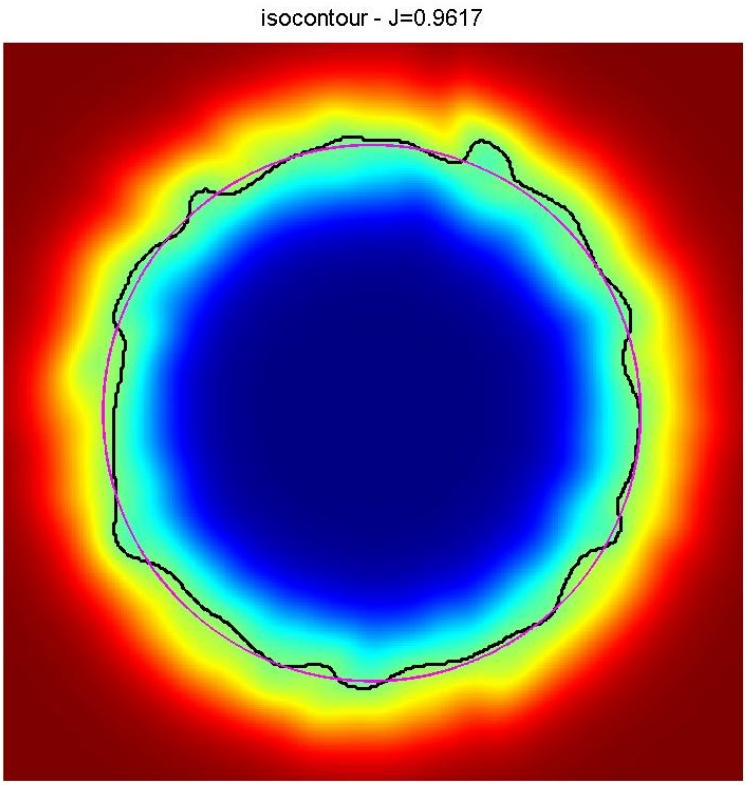


Figura 6.13 - *Visualizzazione del marching squares*

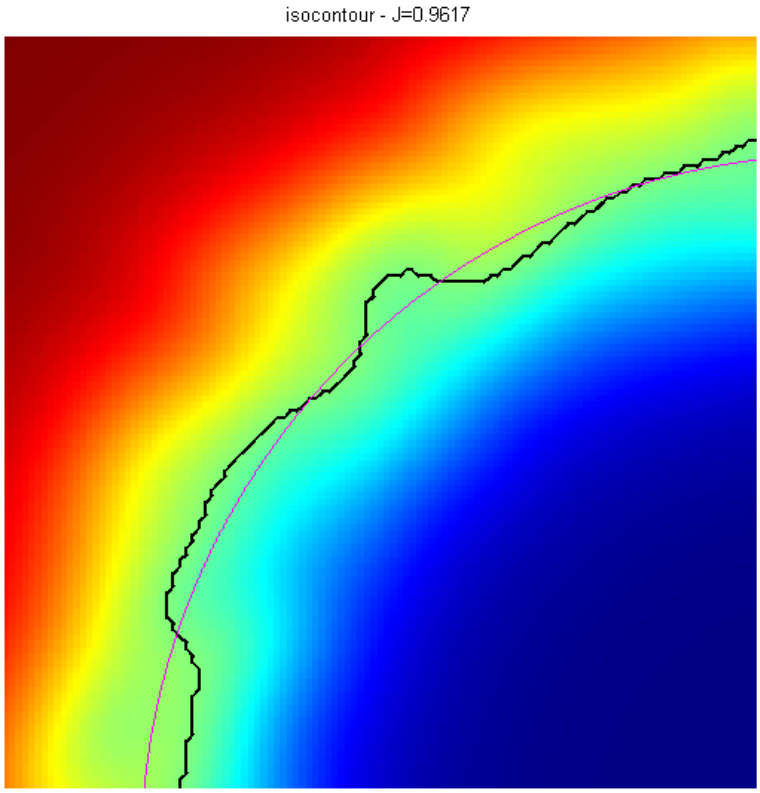


Figura 6.14 - *Particolare della curva. È possibile vedere così la circonferenza disegnata dall'algoritmo (in nero) e la circonferenza di raggio unitario (in magenta).*

6.1.7 Dataset 4: "circ_5_punti_noise_0.000"

Il quarto dataset serve a verificare come si comporta l'algoritmo nel caso in cui siano presenti pochissimi punti e in assenza di rumore. I parametri sono impostati come per il dataset 1 con l'unica differenza che il numero di regole è 14, cioè il massimo numero di regole create. Viene riportato il grafico 3D delle curve di livello (Fig. 6.15) e la ricostruzione della curva tramite l'uso del marching squares (Fig. 6.16). Dalla visualizzazione delle curve di livello si può osservare come non sia più possibile vedere chiaramente la curva di livello 0. In figura 6.17 viene effettuato uno zoom per vedere quanto effettivamente la curva ricostruita tramite l'algoritmo, in nero, si discosti considerevolmente da quella ideale, in magenta. Possiamo notare quindi che l'algoritmo in presenza di pochi punti, anche se non rumorosi, non riesce a ricostruire la curva. In questo caso infatti la curva risultante è più simile a un poligono che a una circonferenza e il J calcolato dalla funzione circPerf2 è pari a 15.0908.

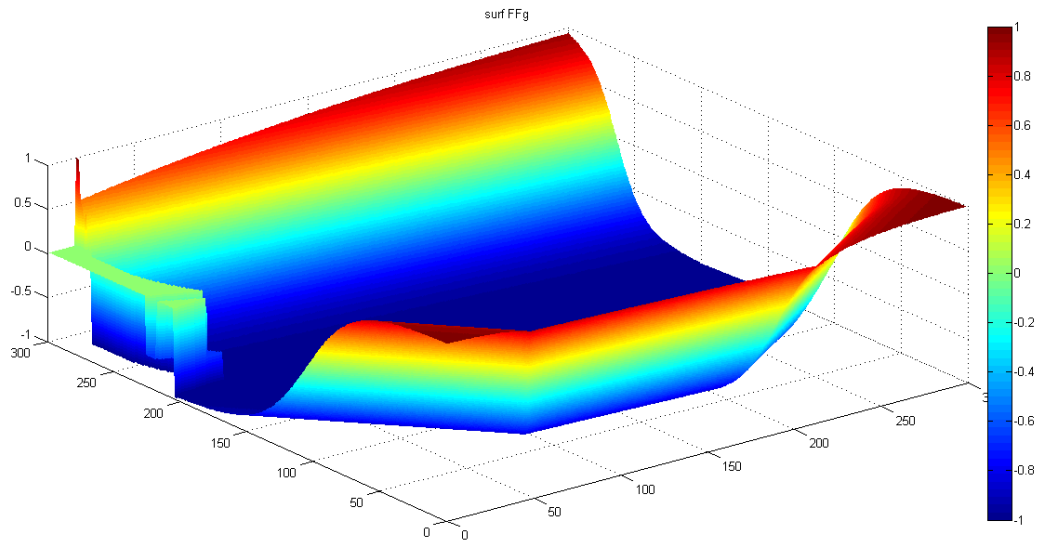


Figura 6.15 - *Grafico 3D delle curve di livello*

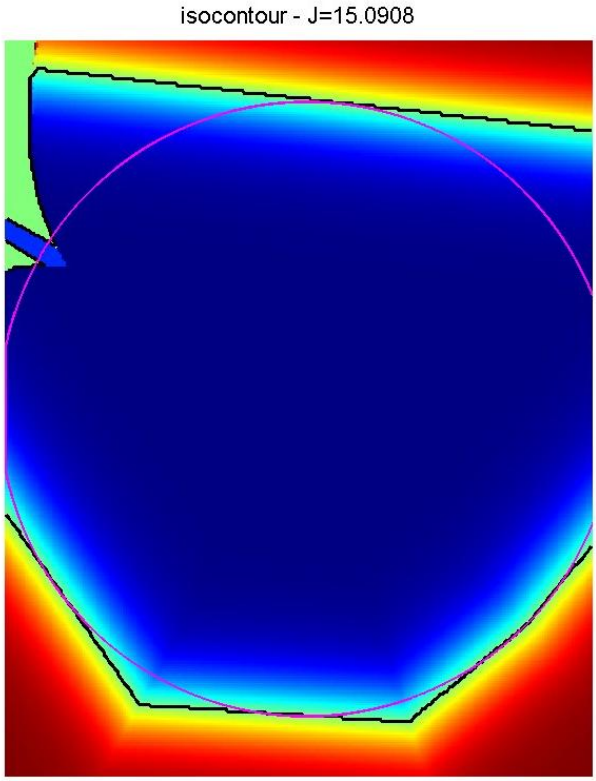


Figura 6.16 – *Visualizzazione del Marching Squares*

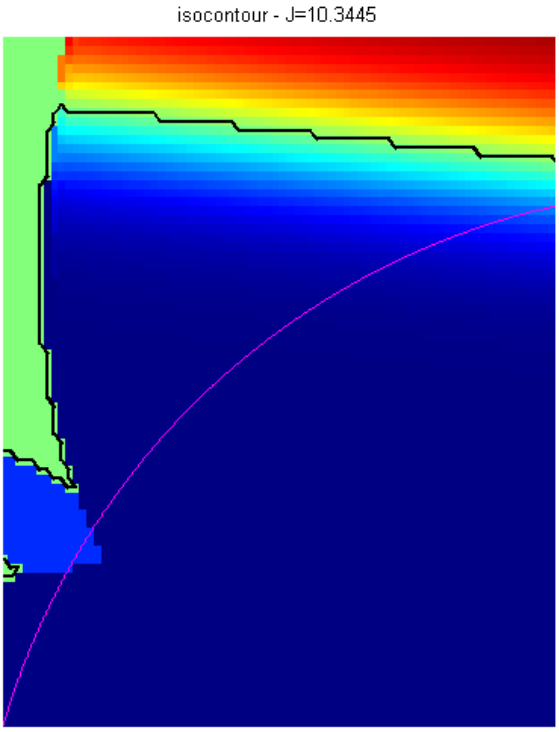


Figura 6.17 - *Particolare della curva. È possibile vedere così la curva ricostruita dall'algoritmo (in nero) e la circonferenza di raggio unitario (in magenta).*

6.1.8 Dataset 5: "circ_5_punti_noise_0.300"

Nel quinto dataset vengono considerati pochi punti ed elevato rumore. I parametri sono impostati come per il dataset 4, quindi con un numero di regole pari a 14, cioè al massimo numero di regole create. Viene riportato grafico 3D delle curve di livello (Fig. 6.18) e la ricostruzione della curva tramite l'uso del marching squares (Fig. 6.19). In figura 6.20 viene effettuato uno zoom e si può facilmente vedere quanto la circonferenza ricostruita tramite l'algoritmo, in nero, sia molto lontana da quella reale, in magenta. Confermiamo così che l'algoritmo in presenza di pochi punti e in più molto rumorosi, non riesce a ricostruire la curva. Anche in questo caso, infatti, la curva risultante è più simile a un poligono irregolare. Il J calcolato dalla funzione circPerf2 è ancora più elevato rispetto al dataset precedente ed infatti è pari a 48,4556.

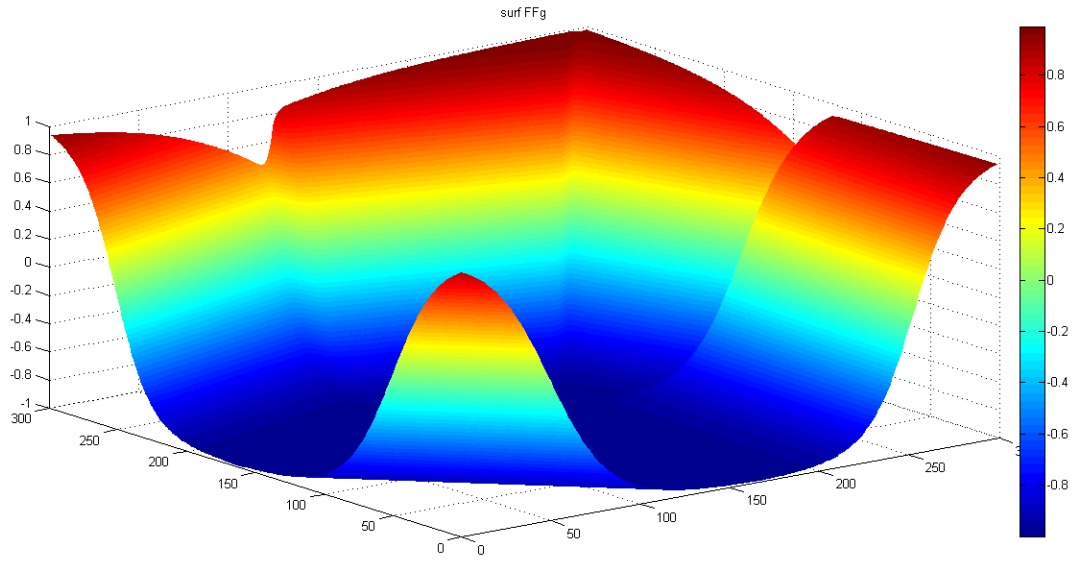


Figura 6.18 - *Grafico 3D delle curve di livello*

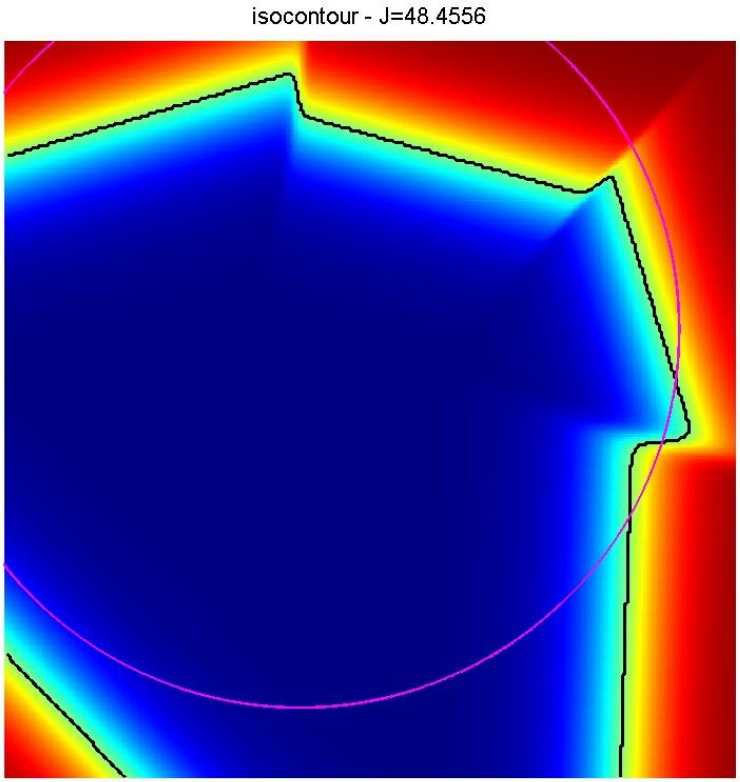


Figura 6.19 - *Visualizzazione del Marching Squares*

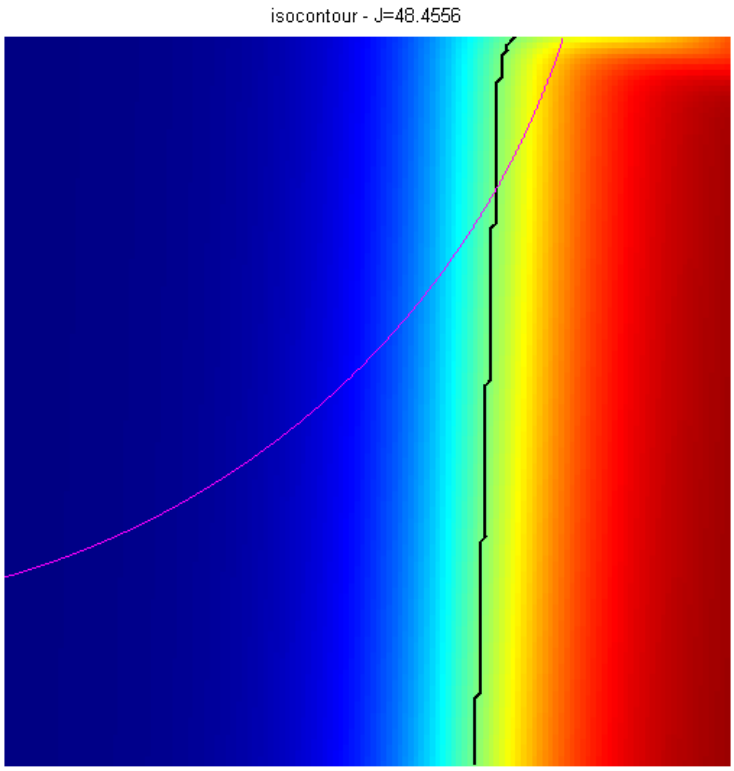
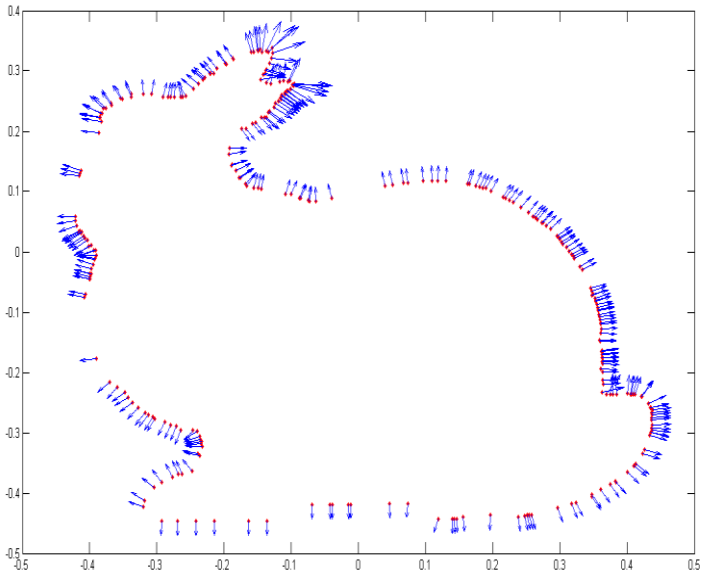


Figura 6.20 - *Particolare della curva. È possibile vedere così la curva ricostruita dall'algoritmo (in nero) e la circonferenza di raggio unitario (in magenta)*

6.1.9 Dataset 6: "slice bunnyhole"

Questo dataset è stato ottenuto selezionando un particolare insieme di punti da bunnyhole.normals, in modo tale da avere una slice del bunny. La funzione che si occupa di effettuare questa selezione di punti è *genBunnnyhole2D* che crea il dataset 6 contenente le coordinate *x* e *y* del bunny e le rispettive normali (Fig. 6.21).



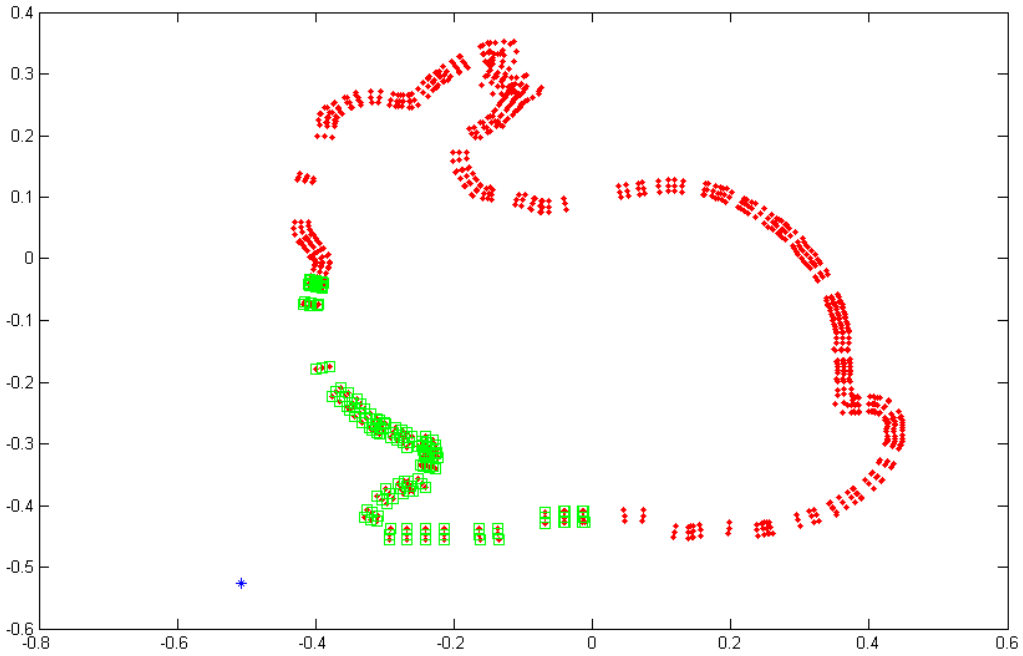
6.21 – *Punti della slice del bunny (in rosso) con normali (in blu).*

In questo caso i parametri impostati sono:

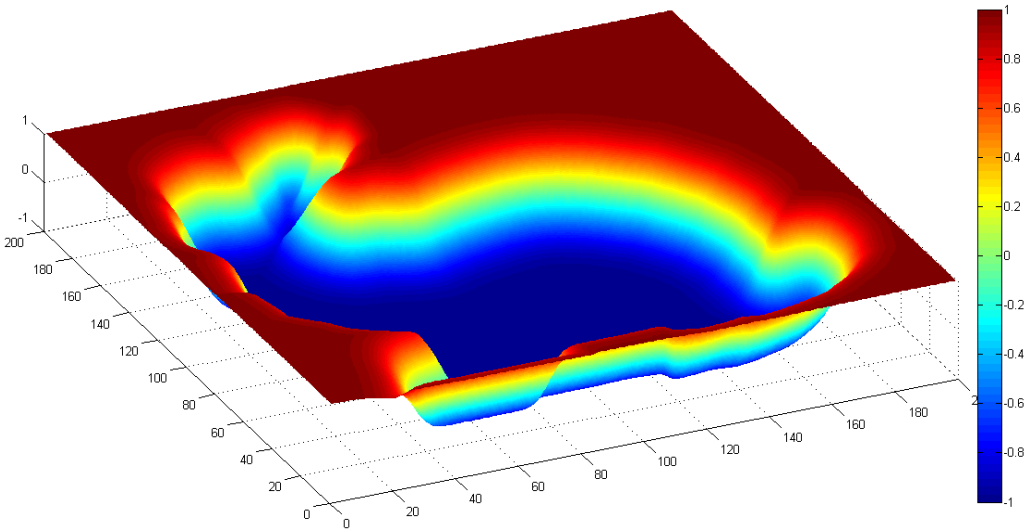
- N = 200;
- SubSamplingFactor = 1;
- K = 150;
- sigma = 15e-3;
- d = 9e-3;
- showLine = true;
- profileTheCode = false;
- knnsearch_handler = @knnsearch_ann_binned;

I risultati ottenuti vengono riportati nelle seguenti figure. In particolare, in figura 6.22, viene visualizzato un esempio di selezione delle regole: l'asterisco blu indica il punto per il quale si stanno valutando le regole, i punti rossi sono i punti della curva e con il quadrato verde vengono evidenziate le regole selezionate. In figura 6.23 abbiamo il grafico

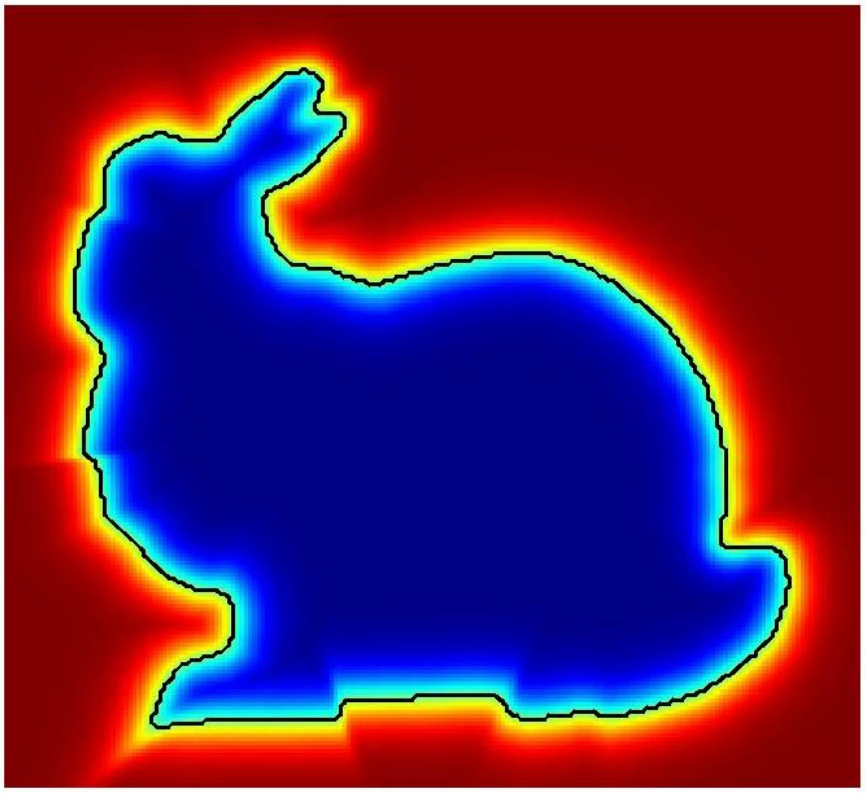
3D delle curve di livello, in cui è facile individuare la curva di livello zero, in verde. Le figure 6.24 e 6.25 invece sono la visualizzazione del marching squares. In questo caso non è stato possibile valutare la bontà della ricostruzione perché non si dispone della curva effettiva.



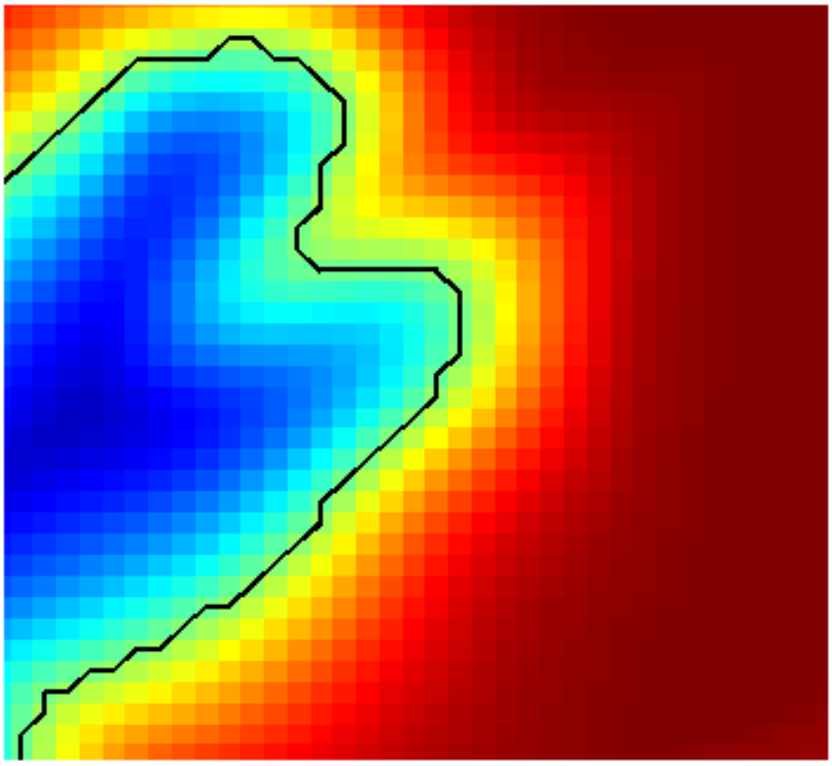
6.22 - *Selezione delle regole: in rosso si hanno le regole, in blu il punto per il quale si valutano le regole e il quadratino verde evidenzia le regole selezionate per tale punto.*



6.23 - *Grafico 3D delle curve di livello*.



6.24 - *Visualizzazione del Marching Squares*



6.25 - *Particolare della curva visualizzata con marching squares.*

7 ESPERIMENTI IN $\Re^3$

In questo capitolo vengono illustrate le prove dell'utilizzo dell'algoritmo nel caso tridimensionale. La funzione è *pcrec3*, implicit surface reconstruction.

7.1 pcrec3

Questa è la funzione che si occupa della ricostruzione di una superficie dati i punti e le normali ai punti.

7.1.1 Dataset per le prove in 3D

Nel caso tridimensionale i dataset sono stati recuperati da internet, dall'indirizzo https://sites.google.com/site/pcsurfacereconstruction/home. I dataset utilizzati sono quelli che contengono anche le normali ai punti e in particolare sono stati presi in considerazione tre dataset: "venus.normals", "bunnyhole.normals" e "knothole.normals". Questi dataset rappresentano tre diverse tipologie di dati:

- Dataset "venus" completo con molti punti (44992 punti);
- Dataset "bunnyhole" con molti punti (341387 punti) e con delle zone in cui non sono presenti punti (buchi);
- Dataset "knothole" con pochi punti (9785 punti) e con buchi.

Nelle seguenti figure si possono vedere i punti dei tre dataset descritti.

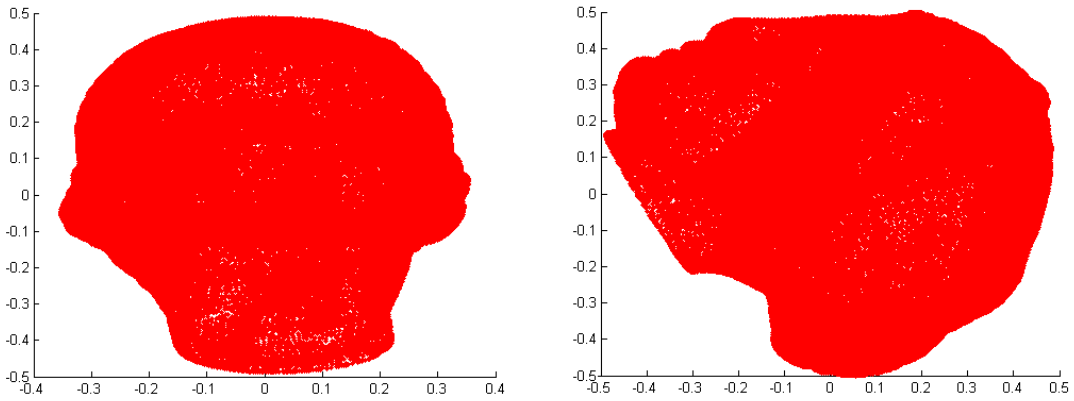


Figura 7.1 – *Dataset "venus"*

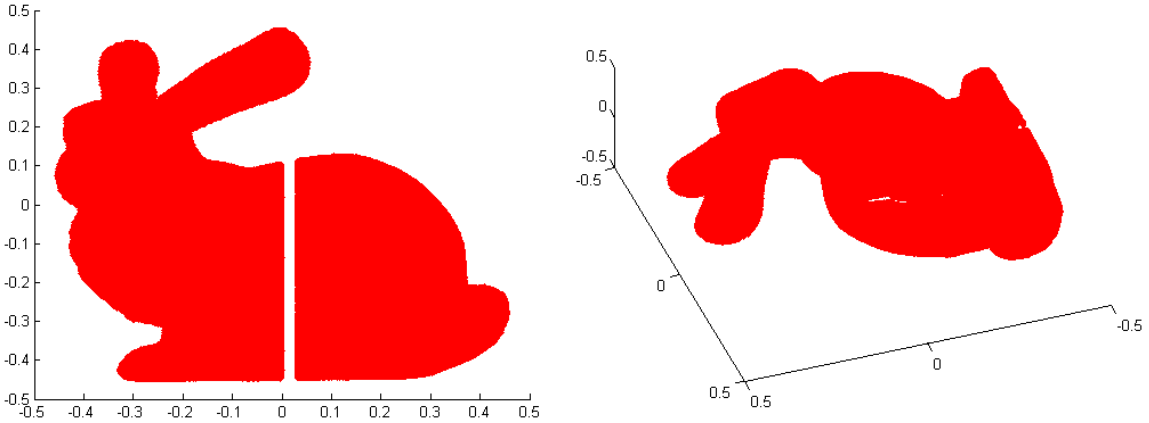


Figura 7.2 – *Dataset bunnyhole*

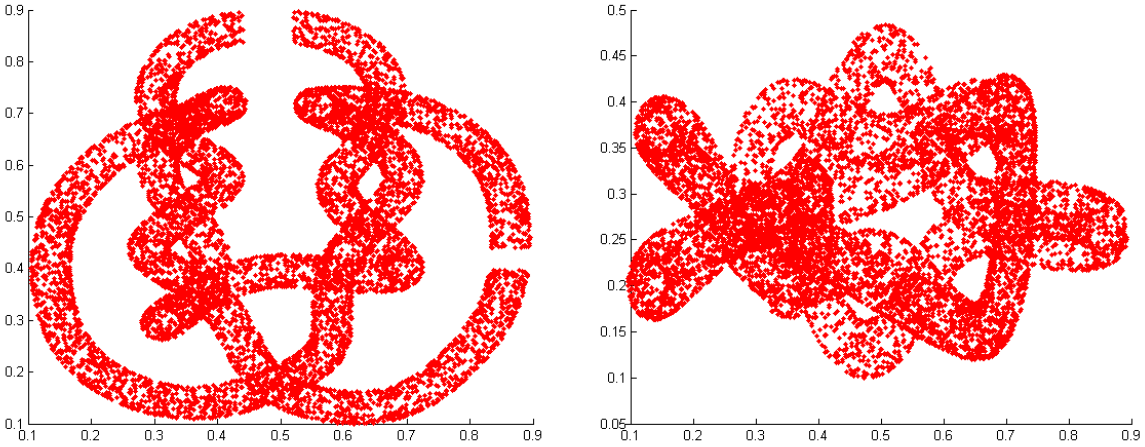


Figura 7.3 – *Dataset knothole*

La figura 7.1 mostra il primo dataset utilizzato, venus, che contiene molti punti; infatti in figura l'intensità del colore dimostra la presenza di molti punti e l'assenza di zone bianche dimostra che il dataset è completo.

La figura 7.2 mostra il secondo dataset utilizzato, bunnyhole, che contiene molti punti ma, al contrario del precedente, presenta delle zone in cui non sono presenti punti; infatti in figura l'intensità del colore dimostra la presenza di molti punti e la presenza di alcune zone bianche dimostrano che il dataset non è completo.

La figura 7.3 mostra il terzo dataset utilizzato, knothole, che presenta delle zone vuote e in più contiene pochi punti; infatti in figura il colore è meno intenso e sono facilmente visibili le zone con assenza di punti.

7.1.2 Parametri

La funzione *pcrec3* ha i seguenti parametri:

- dataname: nome del file (senza l'estensione ".normals")
- N: grandezza della griglia
- subSamplingFactor: fattore usato per il campionamento dei punti
- k: numero di regole utilizzate per l'inferenza
- sigma: deviazione standard delle funzioni di appartenenza gaussiane
- d: distanza dalla curva dei punti aggiuntivi
- showLine: quando TRUE va in esecuzione il Marching Squares
- profileTheCode: quando TRUE va in esecuzione il profiler
- knnsearchHandler: funzione che college alla funzione k-NN search.

Gli esperimenti riportati nei successivi paragrafi sono stati creati tramite un main *(main3D.m)* che manda in esecuzione la funzione *pcrec3* per i dataset presentati nel paragrafo 7.1.1, impostando i seguenti valori dei parametri:

- N = 200;
- subSamplingFactor = 1;
- k = 50;
- sigma = 15e-3;
- d = 1e-2;

- showSurface = true;
- profileTheCode = false;
- knnsearchHandler = @knnsearch_ann_binned;

7.1.3 Dataset "venus"

Il primo dataset proposto per il 3D è un dataset completo, cioè contiene molti punti distribuiti su tutta la superficie. L'algoritmo, come per il 2D, una volta caricati i punti e le loro normali, crea i punti ausiliari interni ed esterni alla superficie. Quindi tramite il sistema TS di ordine 0 crea il modello e tramite l'inferenza fuzzy le regole. Si passa poi alla ricostruzione e visualizzazione della superficie utilizzando la tecnica del *Marching Cubes*. In figura 7.4 si ha la visualizzazione della superficie ricostruita dall'algoritmo, in figura 7.5 si sovrappongono a tale superficie i punti originali del dataset. Si può notare come l'algoritmo riesca ad approssimare anche le zone in cui non sono presenti i punti, ad esempio alla base del collo.

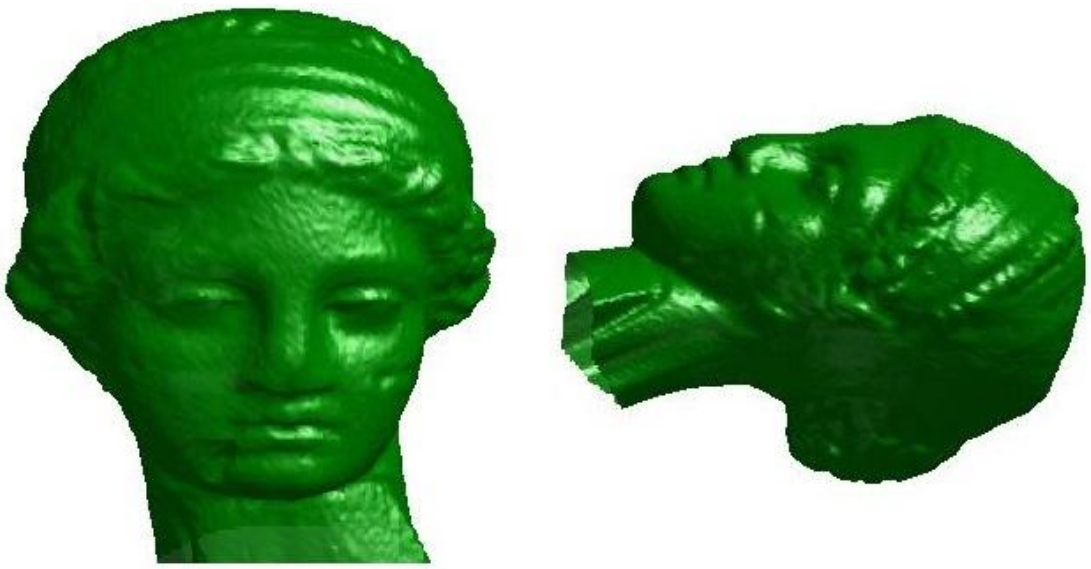


Figura 7.4 – *Visualizzazione della superficie di venus con la funzione isosurface*

Figura 7.5 – *In verde si ha la superficie di venus ricostruita della figura 7.4, in rosso sono stati sovrapposti i punti originali (visione frontale).*

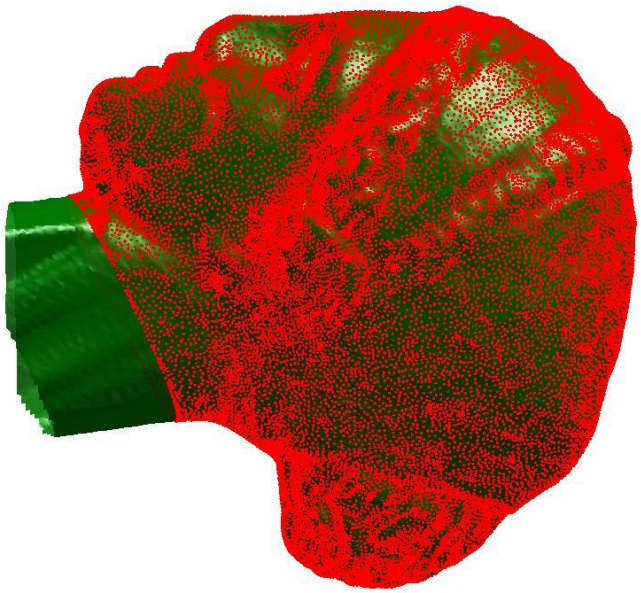


Figura 7.6 - *In verde si ha la superficie di venus ricostruita della figura 7.4, in rosso sono stati sovrapposti i punti originali (visione laterale)*

7.1.4 Dataset "bunnyhole"

Il secondo dataset proposto per il 3D è un dataset con un numero elevato di punti ma che contiene dei buchi, delle zone in cui i punti non sono presenti. I parametri sono impostati come per il primo dataset. L'algoritmo anche in questo caso riesce a ricostruire abbastanza bene la superficie. In figura 7.7 si ha la visualizzazione della superficie ricostruita dall'algoritmo, in figura 7.8 e in figura 7.9 si sovrappongono a tale superficie i punti originali del dataset. In queste figure è particolarmente visibile come l'algoritmo riesca a ricostruire la superficie anche in presenza di buchi.

Figura 7.7 - *Visualizzazione della superficie di bunnyhole con la funzione isosurface*

Figura 7.8 - *In verde si ha la superficie di bunnyhole ricostruita della figura 7.7, in rosso sono stati sovrapposti i punti originali (visione frontale).*

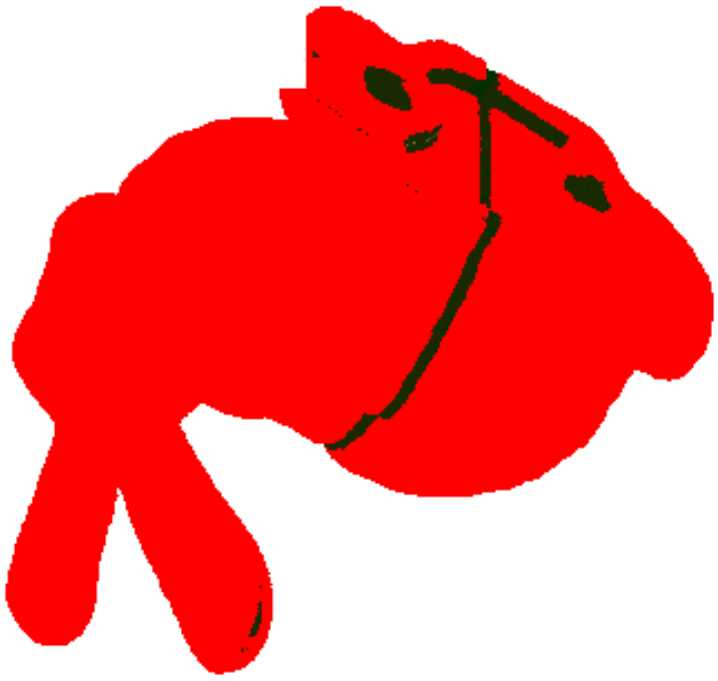


Figura 7.9 - *In verde si ha la superficie di bunnyhole ricostruita della figura 7.7, in rosso sono stati sovrapposti i punti originali (visione laterale).*

7.1.5 Dataset "knothole"

Il terzo dataset proposto per il 3D è un dataset con un numero molto basso di punti e che contiene dei buchi, delle zone in cui i punti non sono presenti. I parametri sono impostati come per il primo dataset. L'algoritmo anche in questo caso riesce a ricostruire abbastanza bene la superficie. In figura 7.10 si ha la visualizzazione della superficie ricostruita dall'algoritmo, in figura 7.11 e in figura 7.12 si sovrappongono a tale superficie i punti originali del dataset. In queste figure si può vedere come l'algoritmo riesca a ricostruire la superficie anche con questo tipo di dataset, cioè in presenza di molti buchi e con a disposizione pochi punti.

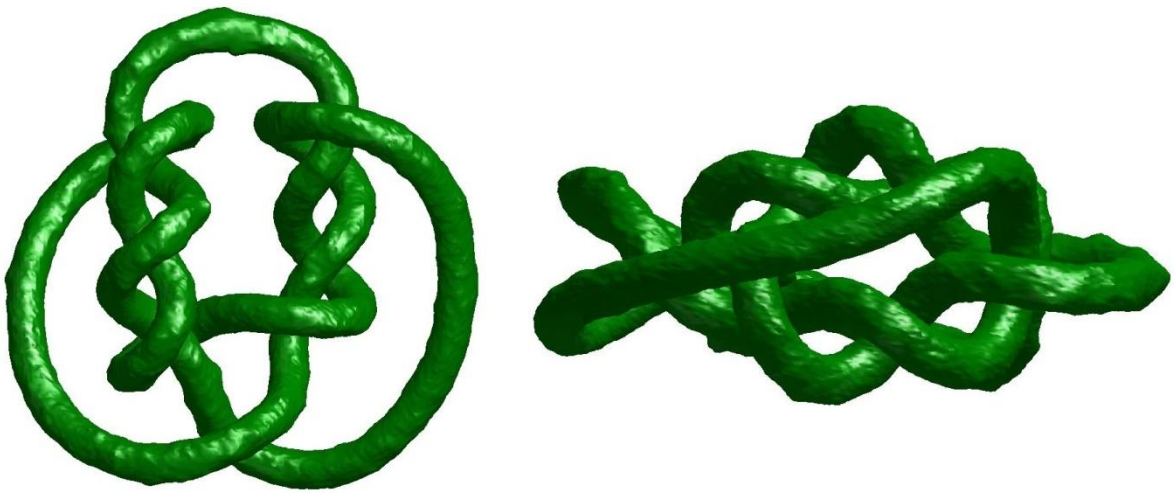


Figura 7.10 - *Visualizzazione della superficie di knothole con la funzione isosurface*

Figura 7.11 - *In verde si ha la superficie di knothole ricostruita della figura 7.10, in rosso sono stati sovrapposti i punti originali (visione frontale).*

Figura 7.12 - *In verde si ha la superficie di knothole ricostruita della figura 7.10, in rosso sono stati sovrapposti i punti originali (visione laterale).*

8 CONCLUSIONI

Con questo lavoro di tesi è stato dimostrato per la prima volta che è possibile ricostruire delle superfici utilizzando sistemi a regole fuzzy, purchè vengano presi opportuni accorgimenti. Infatti, nel corso degli esperimenti in due e tre dimensioni è presto emersa la necessità di utilizzare una tecnica di inferenza più efficiente, per poter ricostruire superfici composte da un elevato numero di punti. E' stata introdotta così per la prima volta in letteratura, la *Truncated Fuzzy Inference*, che è risultata una valida soluzione sia per ridurre significativamente la complessità computazionale sia l'occupazione di memoria. Rimangono tuttavia numerosi aspetti da investigare più in profondità, quali la robustezza del metodo, l'utilizzo di metodi più efficienti di rendering rispetto al marching squares/cubes, ecc… Anche il porting dell'attuale codice sorgente su GPU è di sicuro interesse, perché potrebbe ridurre drasticamente i tempi di calcolo.

BIBLIOGRAFIA

[1] PC Surface Reconstruction. URL: https://sites.google.com/site/pcsurfacereconstruction

[2] Wikipedia URL: www.wikipedia.org

[3] Algorri, M.E., Schmitt, F.: Surface reconstruction from unstructured 3D data. In: Computer Graphics Forum, vol. 15, pp. 47–60 (1996)

[4] Atkenson, G.C., Moore, A.W., Schaal, S.: Locally weighted learning. Artificial Intelligence Review11, 11–73 (1997)

[5] Bajaj, C.L., Bernardini, F., Xu, G.: Automatic reconstruction of surfaces and scalar fields from 3D scans. In: Proceedings of the ACM SIGGRAPH Conference on Computer graphics and interactive techniques, pp. 109–118 (1995)

[6] Dreyfus, G.: Neural networks: methodology and applications. Springer (2005)

[7] Hoppe, H., Derose, T., Duchamp, T., Mc-Donald, J., Stuetzle, W.: Surface reconstruction from unorganized points. In: Proceedings of the ACM SIGGRAPH Conference on Computer graphics and interactive techniques), vol. 26, pp. 71–78 (1992)

[8] Liang, J., Park, F., Zhao, H.: Robust and efficient implicit surface reconstruction for Point Clouds based on convexified image. Journal of Scientific Computing, Vol. 54 (2-3), pp 577-602, 2013.

[9] Lorensen, W.E., Cline, H.E.: Marching cubes: a high resolution 3D surface construction algorithm. ACM SIGGRAPH Computer Graphaphics Newsletter 21(4), 163–169 (1987)

[10] Mencl, R., Muller, H.: Interpolation and approximation of surfaces from three-dimensional scattered data points. In: Proceedings of the Eurographics 98 Conference, pp. 51–67 (1998)

[11] Poggio, T., Girosi, F.: Network for approximation and learning. Proceedings of the IEEE78, 1481–1497 (1990)

[12] Weiss, S., Indurkhya, N.: Optimized rule induction. IEEE Expert8(6), 61–69 (1993)7

RINGRAZIAMENTI

Desidero innanzitutto ringraziare la Prof. Beatrice Lazzerini per aver sostenuto questa tesi e l'Ing. Marco Cococcioni per la disponibilità e le numerose ore dedicate al mio lavoro, per il sostegno, per la pazienza e per gli insegnamenti ricevuti.

Ringrazio i miei genitori per il sostegno morale ed economico…grazie per avermi dato questa possibilità ma soprattutto per aver sopportato i miei alti e bassi.

Un grazie va anche ai miei due fratelli, per avermi aiutato quando non ce la facevo più e aver festeggiato ed esultato con me ad ogni esame…ognuno a modo suo!!! Gabriele con le sue lunghe chiacchierate e Valerio invece con i suoi giretti in macchina a ridere e scherzare.

Grazie anche a quello strano ingegnere conosciuto il primo anno (al secondo giorno di lezione, perché io ho saltato il primo giorno fin da subito!) che mi ha accompagnato in tutti questi anni…Daniele grazie mille per tutto, per il tuo aiuto e il tuo sostegno prima di ogni prova, un po' meno per tutte le volte che finito un esame hai esclamato "Bene ora ti tocca fare anche quest'altro!!!". Grazie anche per le innumerevoli e bellissime esperienze che mi hai fatto vivere in questi anni universitari, viaggi soprattutto e avventure di ogni tipo… (e questa volta niente neve prima della laurea!) e che spero continueranno anche dopo!

Grazie anche ai nonni che mi hanno sempre appoggiata nelle mie scelte e sostenuta nei momenti critici.

Un grazie anche alle mie amiche "dai tempi antichi", la Piccia, Mariangela, Laura, Lavinia, Roberta, Betty e Sara; ai miei amici delle scuole superiori, Camboino, Pariccio, Maggiolino e Urso; alle mie carissime ex coinquiline Manuela e Damiana…e anche alle mie nuove coinquiline Rossella e Debora.

E infine non può mancare un grazie a tutte le persone che vicine o lontane mi sono sempre state accanto e mi hanno accompagnata in questo lungo viaggio…