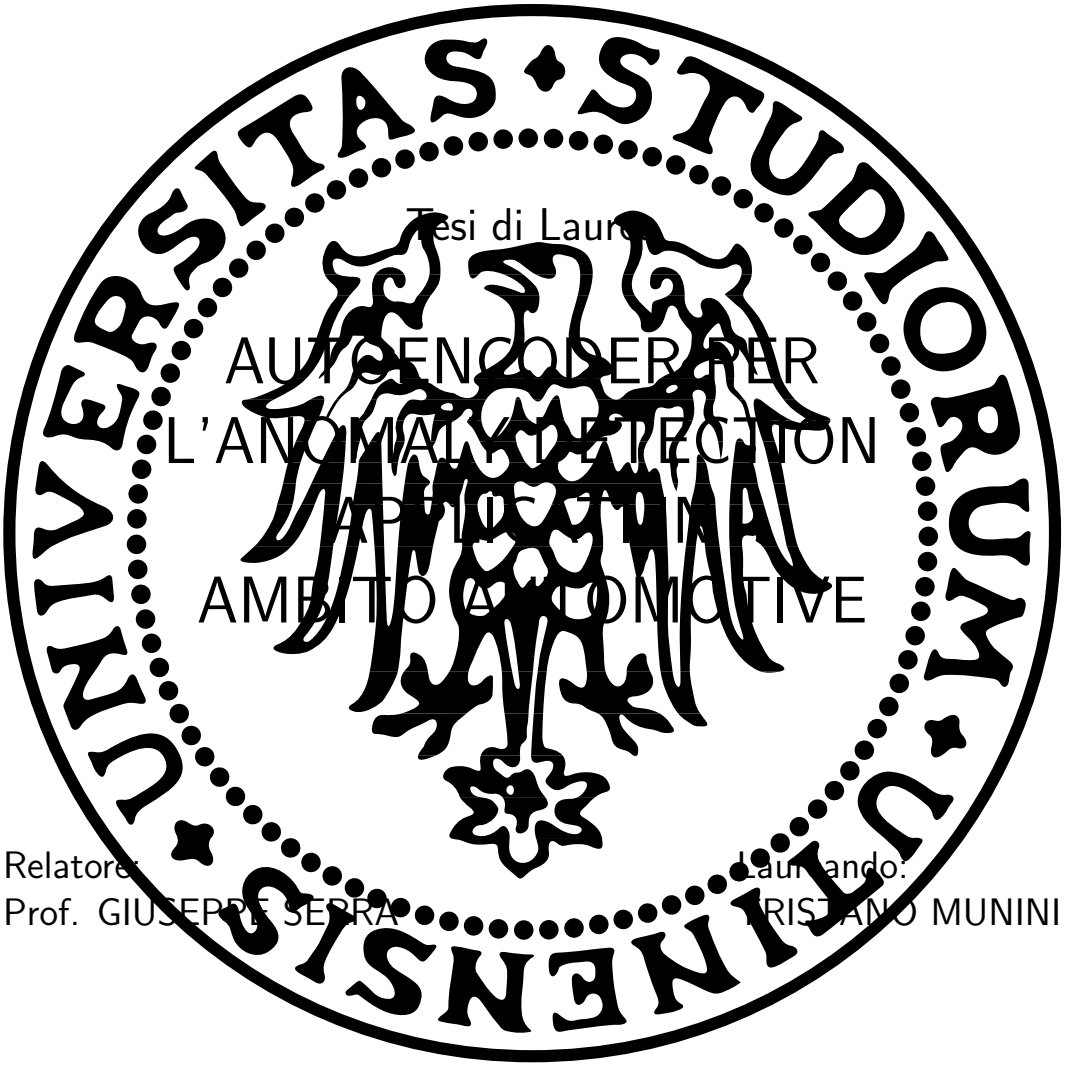
UNIVERSITÀ DEGLI STUDI DI UDINE

Dipartimento di Scienze Matematiche, Informatiche e Fisiche

Corso di Laurea Triennale in Informatica



Tesi di Laurea

AUTOENCODER PER L'ANOMALY DETECTION APPLICATA IN AMBITO AUTOMOTIVE

Relatore:
Prof. GIUSEPPE SERRA

Laureando:
CRISTIANO MUNINI

ANNO ACCADEMICO 2018-2019

Ai miei genitori
per non avermi tagliato i viveri

Abstract

TODO

Introduzione

In campo automotive, soprattutto negli ultimi anni, si è visto un crescente interesse nei confronti di sistemi di intelligenza artificiale che permettano di supervisionare la qualità dei pezzi prodotti. Assicurare la qualità è un requisito critico perché, in generale, la qualità influenza l'intera vita di un prodotto. Le applicazioni di algoritmi di Machine Learning prima, e di sistemi di Deep Learning poi, si sono dimostrate efficaci, flessibili e resilienti, portando numerosi vantaggi non solo nel campo del controllo automatico ma anche in quello del supporto agli operatori umani, ad esempio. I riferimenti alle applicazioni ben riuscite di sistemi intelligenti crescono di mese in mese ed offrono un ottimo mercato I riscontri In un mercato

TODO Sistemare sopra

TODO Spiegare cosa si intende con machine vision

Ref a

Machine vision Computer vision

Indice

Elenco delle figure

Capitolo 1

Il Problema

In questo documento si affronta il problema di rilevare la presenza di colla all'interno di carcasse per motori elettrici, per mezzo di fotografie digitali. Dopo alcune considerazioni sulla forma e sul fine ultimo di tali pezzi seguirà una descrizione sommaria del processo industriale e delle macchine che manipolano e depositano la colla all'interno dei pezzi, nonché del sistema di acquisizione immagini. Si conclude la sezione con una formalizzazione del problema da risolvere e le metriche con cui valutare la soluzione proposta.

1.1 Le Carcasse per Motori Elettrici

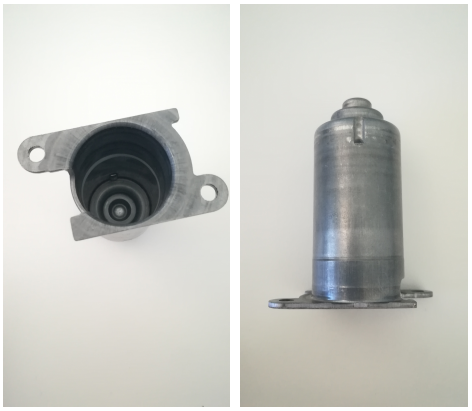


Figura 1.1: Visioni laterale e superiore di una carcassa

Osservando le fotografie in figura, possono essere definite le caratteristiche principali delle carcasse:

- il pezzo ha una struttura cilindrica cava;

- il fondo presenta tre gradini;
- sono presenti due balze (nella foto laterale se ne vede soltanto una), posizionate, una di fronte all'altra, prima del primo gradino;
- i supporti alla bocca della carcassa presentano due fori;

Il pezzo è stato pensato per avvolgere e proteggere motori elettrici. Nello specifico i pezzi in foto, sui quali è stato svolto lo studio, sono carcasse per motori elettrici per tergicristalli. I due fori aiuteranno a fissare con delle viti il pezzo su dei supporti in plastica. Il motore alloggerà interamente nella cavità, nella quale verrà anche depositata della colla. Questa verrà poi cotta in modo che il motore non possa vibrare all'interno della carcassa, evitando che eventuali urti possano danneggiarlo. Dalla foto si può notare che la colla è distribuita in modo da formare un anello ad una altezza di circa $4cm$ dal fondo della carcassa. La colla è stata depositata correttamente se:

1. l'anello non presenta né sbavature né discontinuità;
2. sul fondo non c'è presenza di colla.

In questo documento, come si vedrà nella sezione sul dataset, ci concentreremo esclusivamente sul secondo punto. La presenza di colla sul sul fondo della carcassa causerà il malfunzionamento del motorino dopo un limitato tempo d'utilizzo, molto inferiore al tempo di vita atteso. Per questo motivo è fondamentale che la colla venga depositata correttamente.

1.2 Il Processo di Produzione

Chiarire le modalità con cui le carcasse vengono manipolate, la colla viene depositata e le foto vengono acquisite risulta fondamentale. Senza queste informazioni mancherebbe la base sulla quale costruire ipotesi e considerazioni riguardo le immagini del dataset. Analizzando le condizioni in cui le foto vengono scattate, si definiscono i vincoli ed i confini entro i quali le soluzioni proposte possono considerarsi verosimili ed applicabili al mondo reale.

Le carcasse, già presenti in grandi quantità in magazzino, raggiungono il macchinario e vengono caricate, con la concavità rivolta verso l'alto, su un disco rotante. Ad ogni ciclo macchina la colla, tramite due ugelli, viene depositata simultaneamente su due carcasse distinte. Al contempo due sonde dotate di luce scendono nelle due carcasse su cui la colla era stata depositata il ciclo precedente, fino ad una distanza di circa $3cm$ dal fondo. Le carcasse ritenute conformi procedono lungo un rullo trasportatore, mentre quelle che non idonee vengono scartate.

Vanno precisati vari aspetti. Le carcasse, nonostante siano tutte dello stesso tipo, possono differire per quanto riguarda colore, graffi superficiali, sporco, incrostazioni oppure macchie. Inoltre non vengono orientate tutte allo stesso modo rispetto all'asse verticale: questo ha delle ripercussioni dirette sulle foto raccolte, infatti le due balze non si presenteranno in posizioni fisse.

Il sistema assicura che le foto vengano scattate sempre alla stessa profondità e che la sonda sia centrata rispetto al pezzo (considerando come centro il centro della cavità cilindrica). La distanza fissa è condizione sufficiente per garantire la messa a fuoco di ognuno dei tre gradini sul fondo. Purtroppo non sono stati specificati dei vincoli riguardo l'illuminazione.

Si fa notare che il processo appena descritto viene eseguito da almeno due macchinari distinti, ovviamente questo aggiunge un ulteriore grado di sfida: non si può supporre che i macchinari siano sempre calibrati esattamente allo stesso modo.

In conclusione le foto che saranno da analizzare vengono raccolte da un totale di quattro fotocamere distinte, delle quali si assicura

- con un grado di precisione soddisfacente la distanza dal fondo;
- con un grado di precisione accettabile la centratura delle immagini.

1.3 Gli Obiettivi da Raggiungere

Di seguito sono riportati alcuni dati numerici riguardo i processi appena descritti.

La colla viene depositata su circa 5000 pezzi al giorno, la probabilità che gocce di colla cadano sul fondo delle carcasse è estremamente bassa. Purtroppo non esistono dati numerici esatti ma si stima che il macchinario abbia un *fault rate* di una carcassa al mese o poco più. Questi dati possono essere trasformati in probabilità approssimative osservando che:

colla depositata al mese:	$5000 * 31 =$	155000
colla mal depositata al mese:		2

Quindi la probabilità che il macchinario sbagli e pari allo 0.00001%. Il sistema di intelligenza artificiale deve riconoscere i pezzi non conformi ma soprattutto, tenendo conto della probabilità di cui sopra, deve generare un numero bassissimo di falsi positivi. Ricordano che per falsi positivi (riferiti anche come FP o False Positive) si intendono tutte le carcasse che il sistema considera non conformi ma che in realtà non presentano difetti. Una AI troppo rigida,

che, quando indecisa, propende per scartare il pezzo, inciderebbe negativamente sulla produzione. Si rischierebbe, infatti di creare un enorme danno economico, andando a scartare molte più carcasse del necessario.

Il nostro obbiettivo è quindi quello di generare un numero di falsi positivi che sia inferiore al 2%, cercando di riconoscere più carcasse non conformi possibili.

Capitolo 2

Il Dataset

Innanzitutto si specifica che con la parola Scarto si indica l'immagine di una carcasse che presenta colla sul fondo, pezzo che, quindi, dovrà essere scartato. Invece con la parola Conforme si indica l'immagine di una carcassa nella quale la colla è stata depositata correttamente, quindi con fondo pulito.

TODO recap capitolo In questo capitolo verrà descritto il dataset e quali.

2.1 Problematiche Principali

2.1.1 Dataset Piccolo e Sbilanciato

La prima difficoltà insorge ancora prima di ispezionare le immagini del dataset. Infatti il dataset non solo comprende solamente 1719 immagini ma è anche fortemente sbilanciato:

- 1719 immagini sono Conformi;
- 30 immagini sono Scarti.

A questo punto è corretto chiedersi se le quasi duemila immagini del dataset siano sufficienti ai nostri scopi. Nel campo del *Machine Learning*, ed ancora di più in quello del *Deep Learning*, non è raro che il numero di elementi in un dataset sia dell'ordine delle decine di migliaia se non di quello delle centinaia di migliaia. Basti pensare ai dataset più famosi ed usati:

- MNIST è un dataset molto famoso contenente 70000 cifre disegnate a mano appartenenti a 10 classi in totale. È alla stesso tempo sia il punto di partenza dei principianti, perché di facile manipolazione, sia il campo di prova degli esperti, sul quale vengono allenati nuovi modelli prima di passare a compiti più complessi;

- CIFAR-10 contiene TODO descrivere
- ImageNet contiene TODO descrivere

Ciascuno di questi dataset è stato etichettato[1] a mano. Ciò significa che ad ogni immagine è stata assegnata, a mano, una classe di appartenenza. Prendendo in esempio MNIST, se un'immagine raffigura la cifra 7 allora sarà etichettata con il label *seven* ed apparterrà alla classe di immagini in cui compare la cifra 7. Allo stesso modo le immagini di ImageNet hanno etichette come *dog*, *cat*, *bird*, *car*, *bike*, ecc... Il nostro dataset, come già illustrato, contiene due classi: Conforme e Scarto.

Sembrerebbe che possedere 2000 immagini appena renda impossibile applicare algoritmi di *Machine Learning*. In realtà osservando i Conformi, in figura TODO sono stati riportati alcuni esemplari significativi, ci si accorge che i pezzi sono molto simili. Le differenze principali tra un'immagine e l'altra riguardano la posizione delle balze, la luminosità e le imperfezioni superficiali (graffi, macchie, ...), ciascuna di queste differenze verrà analizzata nel dettaglio tra poco. Quindi, come ci si poteva aspettare essendo pezzi creati meccanicamente, la loro distribuzione è nell'intorno del pezzo progettato. TODO dire meglio. Concludiamo che il numero di Conformi a nostra disposizione è sufficiente

Purtroppo non possiamo dire lo stesso per gli Scarti. Se già la statistica ci lascia sospettare che 30 esemplari non possono ritenersi significativi, allora questo sospetto diventa certezza quando si analizzano le caratteristiche della colla nelle immagini Scarto. Come si vede in figura TODO la colla può presentarsi in forma di gocce più o meno circolari oppure come sbaffi di grossezza e lunghezza variabili. Anche la quantità di superficie coperta dalla colla può variare notevolmente, passando da aree ridotte e localizzate ad aree estese e di conformazioni singolari. Infine notiamo che la posizione del rimasuglio di colla all'interno della carcassa non è in relazione con la posizione delle balze e che la presenza dei gradini sul fondo non la obbliga in alcun modo a scivolare fino al centro.

Per analizzare meglio le modalità con cui potrebbe essere generato uno Scarto si supponga che il macchinario abbia commesso un errore: dall'ugello è uscita un certa quantità di colla in esubero. A seconda della posizione dell'ugello rispetto alla carcassa si può immaginare che la colla raggiunga il fondo in vari modi, proviamo ora ad esplorarne due:

- nel primo caso si immagina che il braccio abbia già depositato l'anello di colla e che si stia allontanando dalla carcassa. La colla in esubero

[1] in inglese *labeled* da *label*, etichetta

cadrebbe sotto forma si goccia fino a raggiungere il fondo del pezzo. Questo potrebbe esse il caso per la figura TODO ref immagine con colla a goccia;

- nel secondo caso si immagina che la colla in esubero faccia parte dell'anello appena depositato e che, a causa delle vibrazioni o di altri fattori simili, coli raggiungendo il fondo della carcassa. Questo potrebbe esse il caso per la figura TODO ref immagine con colla "sbaffata" dal bordo.

Concludiamo che gli esemplari forniti per la classe Scarto descrivono soltanto in modo parziale la distribuzione della classe (TODO dire meglio) e che quindi non possono essere utilizzati per allenare un modello veramente generale. Infatti se, per esempio, venissero usati per il train di una rete convolutiva, di cui poi verrà illustrata brevemente la struttura, si rischierebbe di creare un modello con forte *overfit* rispetto a quelle specifiche macchie di colla fornite. Con il termine overfit si intende TODO.

Prima di proseguire con le prossime problematiche dobbiamo spendere alcune parole per commentare la colorazione delle immagini rispetto ai veri colori delle carcasse e della colla. In figura 1.1 a pagina 1 abbiamo visto che la superficie del pezzo è di colore grigio ma nella foto risulta di colore verdastro. Allo stesso modo anche la colla, in realtà di colore bianco sporco, nella fotografia assume tonalità verdognole. Per certi compiti possedere immagini in falsi colori può risultare problematico ma fortunatamente non è questo il caso: l'importante è che venga mantenuta l'informazione che ci permette di distinguere la colla dalla superficie della carcassa. Come vedremo poi le immagini verranno trasformate in scala di grigi quindi, nonostante sarebbe stato preferibile avere immagini a colori reali, i falsi colori non sono da considerarsi problematici.

2.1.2 Differenze tra Immagini

Ora che abbiamo una visione d'insieme sul dataset possiamo concentrare la nostra attenzione sulle proprietà principali delle immagini. Innanzitutto ogni immagine ha una risoluzione di 896x896 pixel, dimensione che ci permette di esplorare varie possibilità. Ad esempio si può pensare di ridurre l'immagine ad una dimensione tale da: occupare meno spazio in memoria, quindi in RAM durante il training, ed allo stesso tempo di mantenere un livello di dettaglio sufficiente ai nostri scopi, risultando quindi in un boost in velocità di train. TODO dire meglio. Oppure di suddividere l'immagine in quadranti da analizzare singolarmente così da mantenere la qualità dell'immagine originale ma senza dover creare una rete che accetti immagini troppo grandi. Infatti una rete che accetta immagini di grandi dimensioni, solitamente, avrà un numero

di parametri maggiore di una che accetta immagini piccole. Questo porta non solo ad occupare più spazio in memoria ma significa anche che la rete impiegherà più tempo sia in fase di train (più parametri da aggiustare) sia in fase di predizione (più conti da fare). TODO sistemare. Per avere un termine di paragone basti pensare che le immagini di MNIST sono 64x64 pixel mentre quelle di ImageNet di $TODOxTODOpx$.

Osservando nuovamente Figura TODO ci si accorge che le immagini hanno varie proprietà, verranno ora elencate e commentate in ordine crescente di fastidiosità (TODO dire meglio).

- Ogni immagine presenta tre circonferenze concentriche, con centro il centro del pezzo. Ciascuna circonferenza è definita da una transizione da una zona più scura ad una più chiara. Sappiamo che le zone più scure corrispondono alle pareti verticali del pezzo mentre le zone chiare ai tre gradini sul fondo. Questa proprietà non è problematica, anzi potrà essere sfruttata a nostro vantaggio.

- Dato che le immagini vengono raccolte ad una distanza costante dal fondo, la dimensioni delle circonferenze sono fissate e si mantengono coerenti tra le immagini. Anche questa proprietà verrà usata a nostro vantaggio.

- Le due balze sulla parete verticale sono ben visibili e possono presentarsi, sempre una di fronte all'altra, in ogni posizione lungo una circonferenza di raggio pari al raggio della cavità cilindrica. Possono essere considerate un problema in quanto rappresentano informazione superflua e variabile. Ricordiamo che la posizione delle balze non ha alcuna correlazione con la presenza della colla, tanto meno con la sua posizione.

- Le superfici dei pezzi si assomigliano: presentano tutte un effetto chiamato "sale e pepe" con granuli di grandezze e luminosità varia. Questo è un bene perché è una costante ma anche un male perché ognuno ha una particolare disposizione di quella texture. TODO dire meglio. Bisogna prestare particolare attenzione alle macchie scure presenti sul fondo di alcune carcasse. La posizione delle macchie non è fissa, perdipiù anche la loro dimensione è variabile. Queste qualità superficiali non saranno da sottovalutare in fase di elaborazione delle immagini.

- A guardare le immagini sembra che siano centrate in realtà ci siamo accorti che c'erano delle differenze. TODO dire meglio. Il centro dell'immagine non corrisponde con il centro del pezzo. Nonostante la distanza massima tra centro del pezzo e il centro dell'immagine è tale che il fondo della carcassa sia sempre visibile interamente, è preferibile che le carcasse nelle immagini vengano centrate correttamente.

- La variazione di luminosità tra le varie foto è una problematica che dovrà essere assolutamente gestita. Infatti alcune immagini hanno una luminosità così alta da far risultare alcune superfici bianche. Altre immagini invece sono molto più scure, tanto che anche le zone che normalmente rifletterebbero sono illuminate appena.

Ora possiamo elencare le proprietà esclusive degli Scarti, in questo caso sono tutte a nostro favore poiché sono l'informazione con cui si distingue uno scarto da un conforme(TODO dire meglio):

- La colla ha alcune caratteristiche particolari: ha un colore bianco-verde solitamente più chiaro della superficie della carcassa e presenta sempre delle zone con dei riflessi.

- La colla è localizzata. Significa che, se presente, non appare come tante gocce sparse ma come un corpo unico più o meno allungato.

- Tracciando una diametro a piacere ci si accorge che gli Scarti sono sempre asimmetrici, invece i conformi, a meno di piccole differenze superficiali, sono sempre simmetrici. (TODO dire meglio).

TODO manca qualcosa? TODO fare conclusione section/subsection?

2.2 Pre-Processing

Questa sezione è divisa in due parti: nella prima verranno illustrate alcune tra le principali tecniche di *Digital Image Processing* nonché di *Computer Vision*, esponendo i dettagli matematici ed esplorando le loro applicazioni; nella seconda si spiegherà quali di queste tecniche, in che ordine e per quali motivi sono state utilizzate.

Come prima cosa è bene ricordare che con *Digital Image Processing* si intende il modificare immagini digitali per mezzo di algoritmi eseguibili da un calcolatore.

TODO descrivere pre-processing anziché Digital Image Processing?(Il DIP nelle altr sezioni o mai) Il pre-processing è il manipolare le immagini digitali per renderle utilizzabili da altri algoritmi, nel nostro caso si manipoler

Gli algoritmi utilizzati in questi campi hanno precise formulazioni matematiche perché ogni immagine viene rappresentata come una matrice bidimensionale, se in scala di grigi, oppure tridimensionale se a colori. Gli elementi di una matrice bidimensionale appartengono all'intervallo $[0, 255]$, nel quale 0 corrisponde al colore nero mentre 255 corrisponde al bianco. Disponendo una

sopra l'altra tre matrici come quelle appena descritte si ottiene un'immagine a colori: ogni matrice rappresenta uno dei canali principali (Red, Green, Blue da cui il famoso acronimo RGB) dell'immagine. Sia I un'immagine a tre canali (RGB), il colore del pixel in posizione (i, j) è dato dalla tripletta $(I[i, j, 0], I[i, j, 1], I[i, j, 2])$, in cui: $(0, 0, 0)$ indica il colore nero, $(255, 255, 255)$ indica il colore bianco, $(255, 0, 0)$ indica il colore rosso, $(0, 255, 0)$ indica il colore verde, e così via ... Quindi un'immagine avrà un numero finito di elementi, detti *pixel*, il cui numero si può ottenere moltiplicando il numero di colonne della matrice per il numero di righe.

Uno dei vantaggi del rappresentare le immagini come matrici è quello di poter applicare operazioni classiche come somma, sottrazione, prodotto e divisione. Ma la nostra attenzione si concentrerà soprattutto sulle convoluzioni e sulle trasformazioni affini.

Convoluzioni ATTENZIONE QUESTA E' LA CONVOLUZIONE IN UN LAYER CONVOLUTIVO INVECE IN QEUSTA SEZIONE QUA BISOGNA CENTRARE I KERNEL SU OGNI VALORE E SOSTITUIRLO IN OUTPUT CON IL VALORE OTTENUTO NELLA SEZIONE SUGLI AE SPIEGO CHE D'ORA IN POI LE CONVOLUZIONI SONO DA INTENDERE DIVERSAMENTE CIOE' CON QUALITA' DI FEATURE EXTRACTION (NOTA SUL PADDING CHE FA TORNARE LE CONV DEL AE COME QUELLE DA DESCRIVERE QUA)

Nell'ambito del *Image Processing* con convoluzione si intende l'operazione che permette di effettuare, per ogni pixel dell'immagine, una somma pesata tra il pixel e gli elementi a lui vicini. I pesi sono definiti in una matrice, detta *kernel* o filtro, di dimensioni non superiori a quelle dell'immagine di partenza. Solitamente i kernel hanno dimensione $3x3$ o $5x5$. Sfrutteremo ora un esempio per spiegare come viene effettuata una convoluzione, più avanti, quando parleremo del filtro Sobel, verrà illustrato lo pseudocodice che ne formalizza i passaggi. In Figura 2.2 sono illustrate, da sinistra a destra: l'immagine di partenza I, il filtro K e l'immagine risultato Y. Il simbolo $*$ denota l'operazione di convoluzione e non è da confondere con nessun tipo di prodotto. Effettueremo una convoluzione da sinistra a destra e dall'alto verso il basso, ma si fa presente che l'ordine d'esecuzione non modifica il risultato.

- Il primo passo è posizionare una copia di K sopra ad I in modo che l'angolo in alto a sinistra del kernel combaci con quello in alto a sinistra dell'immagine.

- Ora possiamo moltiplicare ogni elemento di I con il rispettivo elemento di K. I valori così ottenuti dovranno essere sommati tra di loro. Effettuando

i conti si osserva che il risultato combacia con il valore in alto a sinistra dell'immagine risultato.

- Il prossimo passo consiste nel far traslare il kernel di una cella a sinistra, calcolare la somma di prodotti rispetto ai nuovi valori ed infine salvare il risultato in Y una cella a destra rispetto a prima.
- Si prosegue in questo modo fino a che l'ultima colonna di K non combacia con l'ultima colonna di I, questo coincide con il riempimento della prima riga in Y.
- Ora bisogna posizionare K in modo che l'elemento nell'angolo superiore sinistro sia sopra all'elemento in I sulla prima colonna della seconda riga. Si effettuano prodotti e somme, salvando il risultato nella prima cella libera della seconda riga in Y.
- Si procede in questo modo finché non si raggiunge l'ultimo elemento dell"immagine di partenza.

L'algoritmo può essere facilmente espresso tramite due cicli *for* facendo attenzione a posizionare il kernel sempre entro i limiti dell'immagine di partenza.

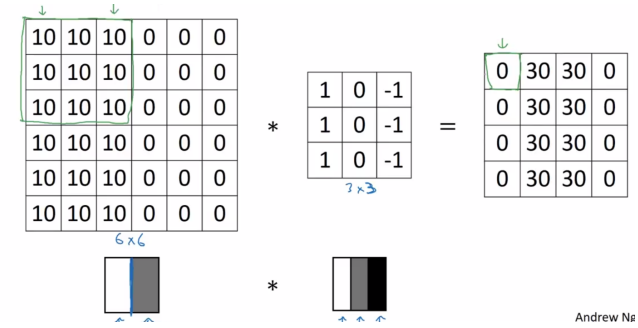


Figura 2.1: TODO Immagine da rifare

Trasformazioni Affini TODO RIEMPIRE DI TEORIA QUA

Come ultima cosa prima di cominciare a descrivere gli algoritmi, si vuole specificare cosa significa *Feature Extraction*.

TODO

Ora verranno introdotti alcuni algoritmi fondamentali nel campo della *Computer Vision*.

TODO aggiungere applicazioni principali per ogni tecnica?

TODO sono da riscrivere meglio.

2.2.1 Passaggio da RGB a GrayScale

La conversione di un'immagine da RGB in scala di grigi è un'operazione estremamente facile, ma rimane comunque alla base di molti algoritmi di image processing. Infatti per molti compiti l'informazione sul colore non è necessaria. Il modo più semplice per combinare i tre canali RGB in un unico canale è quello di fare la media dei valori pixel per pixel:

$$Y = (R + G + B)/3 \tag{2.1}$$

In questo modo ogni canale partecipa allo stesso modo. Però sappiamo che l'occhio umano è più sensibile ai colori verdi, quindi potrebbe essere preferibile dare più importanza al secondo canale:

$$Y = 0.299 * R + 0.587 * G + 0.114 * B \tag{2.2}$$

I pesi sono stati definiti nello standard CCIR 601. Il risulato di quest'ultimo calcolo è rappresentato in Figura 2.2.1.

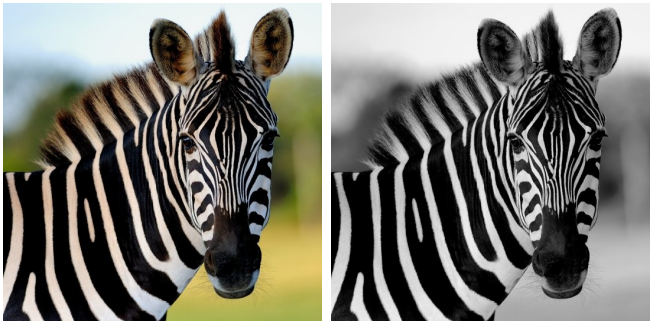


Figura 2.2: A sinistra l'immagine originale. A destra la versione in scala di grigi

2.2.2 Masking

La tecnica del masking permette di nascondere parti di immagine a cui non siamo interessati. Abbiamo bisogno di una maschera binaria, nella quale ogni pixel appartiene all'insieme $\{0, 1\}$, che solitamente viene generata a mano, ed ovviamente dell'immagine che si vuole mascherare. La maschera deve avere le stesse dimensioni dell'immagine di partenza. L'operazione consiste nell'effettuare un AND logico, pixel per pixel, tra l'immagine e la maschera. Così facendo tutti i pixel dell'immagine di partenza corrispondenti a zone di valore 0 della maschera verranno impostati a 0, diventando quindi neri. Il resto dell'immagine rimane con i colori originali.

Nell'esempio in Figura 2.2.2 si è deciso di rimuove l'informazione relativa allo sfondo.

Figura 2.3: Da sinistra a destra: immagine originale, maschera binaria, risultato del masking

2.2.3 Traslazioni e Rotazioni

Come abbiamo accennato prima, le trasformazioni affini ci permettono di effettuare traslazioni e rotazioni alle immagini.

TODOOOOOOO

Con la matrice per il cambio di base hhkk

Figura 2.4: TODO cambiare immagini

2.2.4 Image Cropping and Resizing

Quando si effettua un *image cropping* si ritaglia una porzione dell'immagine, che viene chiamata ROI (*Region Of Interest*), sulla quale vogliamo concentrare la nostra attenzione. Dato che ogni immagine è rappresentata con una matrice, una ROI non sarà nient'altro che una matrice di dimensioni minori in cui sono stati copiati i valori dell'area interessata. Una matrice di questo tipo viene anche chiamata *view*.

Con l'*image resizing* si aumentano (o diminuiscono) le dimensioni di un'immagine. Nel primo caso, poiché si vuole aumentare il numero di pixel dell'immagine finale,bisognerà utilizzare tecniche di upsampling ed interpolare i dati a disposizione per generarne di nuovi che siano verosimili.

Uno fra gli algoritmi più semplici è Nearest-Neighbor Interpolation: il pixel che deve essere aggiunto ottiene il valore del pixel a lui più vicino. Un criterio di scelta dovrà essere definito nel caso in cui ci siano più pixel alla stessa distanza ma con valori differenti. Un criterio possibile è quello di assegnare al nuovo pixel sempre il valore del pixel in alto a sinistra.

Una tecnica leggermente più complessa, ma che fornisce risultati soddisfacenti nella maggior parte delle occasioni, è l'interpolazione bilineare. Con questa tecnica si effettuano, in cascata, due interpolazioni lineari, una orizzontale ed una verticale. Con l'interpolazione bilineare i nuovi pixel si ottengono come media dei valori noti, pesata rispetto allo loro distanza dal pixel che si vuole colorare. In questo modo si ottengono immagini con cambi di colore più dolci.

TODO Parentesi sulle formule? Linear Interpolation per poi spiegare Bilinear Interpolation?

Nel caso in cui si voglia ridurre le dimensioni dell'immagine si dovranno usare tecniche di *downsampling* ed *anti-aliasing*. Il downsampling permette di selezionare un numero limitato di pixel che poi verranno usati per colorare la matrice di dimensione ridotta. Applicare soltanto questa tecnica può portare alla creazione di artefatti sintetici nell'immagine risultato. Significa che l'immagine ridotta potrebbe contenere gruppi di pixel di colori sbagliati. Sfruttando tecniche come l'anti-aliasing si può evitare, o quantomeno limitare, la creazione di tali artefatti. Un *low-pass filter* è un tipo di filtro che smorza tutti i valori al di sopra di una certa soglia, mentre lascia passare tutti i valori minori. Nel campo del *digital image processing* vengono utilizzati filtri di *blur* applicati tramite convoluzione. Il termine *blur* o *smooth* indicano applicare un effetto sfuocato che tende a rendere più dolci le transizioni da un colore all'altro, andando quindi anche a ridurre valori troppo alti (o troppo bassi) avvicinandoli a valori più probabili.

2.2.5 Histogram Equalization

Questa tecnica permette di "aggiustare" il contrasto di un'immagine sfruttandone l'istogramma. Il contrasto è definito come la differenza in intensità luminosa e colore che permette di distinguere gli oggetti.

In Figura 2.2.5 sono stati riporta un'immagine a basso contrasto e l'immagine risultato dopo l'applicazione della *histogram equalization*. Sotto ogni immagine si possono osservare i relativi istogrammi: sull'asse delle x abbiamo ogni possibile valore di un pixel, quindi da 0 a 255; sull'asse delle y è riportato il numero di occorrenze di quel colore nell'immagine. Notare che l'immagine in input deve essere in scala di grigi.

Vediamo ora come possiamo formulare matematicamente la costruzione dell'istogramma e la sua manipolazione. Siano X l'immagine di partenza ed Y l'immagine risultato. Facendo riferimento a quanto scritto nel documento (TODO AGGIUNGERE REF A DOC) l'istogramma può essere definito come:

$$p_n = \frac{\#\text{ di pixel di colore } \quad n}{\#\text{ totale di pixel}} \quad \text{con} \quad n \in [0, 255]$$

Poiché p_n descrive la probabilità che un pixel, scelto a caso dall'immagine, abbia valore n, possiamo considerare X una variabile casuale discreta in $[0, 255]$. Quindi X avrà funzione di ripartizione, tracciata in nero nel grafico, pari ad:

$$F_X(x) = \sum_{n=0}^{x} p_n$$

Noi vorremmo che la differenza di colore tra i pixel fosse più netta, così da aumentare il contrasto dell'immagine. Per raggiungere i nostri scopi possiamo ridistribuire equamente i valori di X nell'intervallo dei valori possibili:

$$T(x) = \text{floor}(255 * F_X(x))$$

dove floor() arrotonda verso il basso all'intero più vicino. In questo modo possiamo ottenere la variabile casuale trasformata $Y = T(X)$, ossia l'immagine equalizzata. Sostanzialmente abbiamo ricolorato ogni pixel dell'immagine di partenza con colori più distanti tra loro.

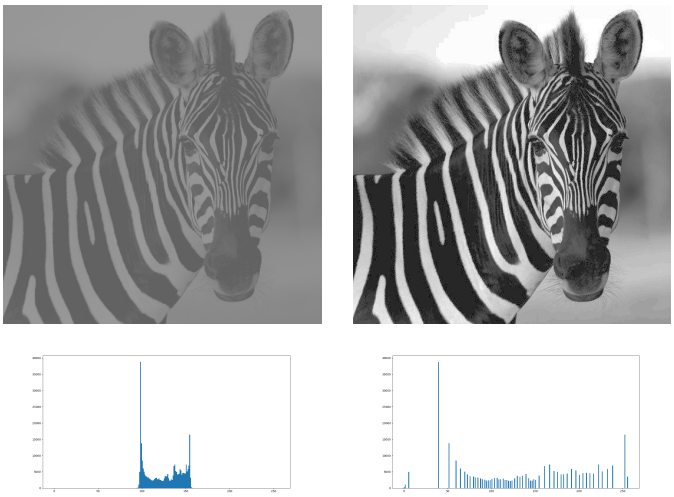


Figura 2.5: TODO rifare caption e grafici

2.2.6 Gaussian Blur

Conosciuto anche come *Gaussian smoothing*, il *Gaussian blur* permette di sfocare un immagine sfruttando una convoluzione con kernel gaussiano. L'immagine così ottenuta risulta meno nitida, con meno dettagli e, quindi, con meno rumore. L'obbiettivo principale di questa tecnica è mantenere soltanto l'informazione caratterizzante, rimuovendo quella non necessaria od anomala. Si può dimostrare che il *Gaussian blur* è un *low-pass filter*.

Si ricorda che la funzione di Gauss ad un parametro è

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

Si dimostra che la funzione di Gauss a due parametri equivale al prodotto di due funzioni come quella appena definita, in formule

$$\begin{aligned} G(x,y) &= G(x)G(y) \\ &= \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \end{aligned} \tag{2.3}$$

in cui x ed y sono le distanze dagli assi di riferimento, mentre σ è la deviazione standard. In termini di calcolo computazionale ciò può essere sfruttato eseguendo due computazioni lineari, rispetto alle dimensioni dell'immagine e del kernel, anziché una quadratica.

$O\left(w_{\text{kernel}} w_{\text{image}} h_{\text{image}}\right) + O\left(h_{\text{kernel}} w_{\text{image}} h_{\text{image}}\right)$

$O\left(w_{\text{kernel}} h_{\text{kernel}} w_{\text{image}} h_{\text{image}}\right)$ TODO correggere e dire meglio.

La matrice sottostante rappresenta un filtro gaussiano quadrato di lato 7 con $\sigma = 2$.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Ricordiamo che durante una convoluzione si effettua la somma di prodotti elemento per elemento, ossia una media pesata in cui i pesi sono definiti nel kernel. Dato che i valori al centro del filtro sono più grandi di quelli ai bordi, daremo maggior peso ai pixel nell'intorno dell'elemento su cui il kernel viene centrato. Sappiamo che all'aumentare di σ cresce il raggio alla base della campana di Gauss, questo significa che verrà data sempre più importanza ai pixel distanti. Ciò farà risultare l'immagine in uscita molto più sfocata.

In Figura 2.2.6 è riportata un'immagine prima e dopo l'applicazione del filtro appena descritto, si nota come i dettagli più piccoli sono stati rimossi, mentre l'oggetto dell'immagine rimane distinguibile.

Figura 2.6: TODO rifare caption

Va fatta un'ultima considerazione: COSA SUCCEDE SE APPLICO DUE VOLTE LO STESSO FILTRO? Se si applica più volte uno stesso filtro gaussiano si ottiene lo stesso risultato che si otterrebbe dopo un'unica applicazione di un filtro TODO completare

2.2.7 Bilateral Filter

TODO

2.2.8 Sobel Operator

Prima di descrivere cosa sia il *Sobel operator*, detto anche *Sobel filter*, bisogna dare una definizione di gradiente. Nel calcolo vettoriale, il gradiente è la generalizzazione della derivata. La derivata di una funzione ad una variabile associa ad ogni punto uno scalare, mentre il gradiente di una funzione f a più variabili associa ad ogni punto un vettore multidimensionale. Quest'ultimo è composto dall'insieme delle derivate parziali di f nel punto considerato. Il gradiente rappresenta la pendenza della tangente al grafico della funzione in un punto. La sua direzione indica il più grande incremento della funzione mentre la magnitudine è il tasso d'incremento.

Possiamo pensare un'immagine (in scala di grigi) come una funzione a due variabili che associa ad ogni punto (x, y) un valore in $[0, 255]$. Il gradiente dell'immagine sarà composto da vettori direzionati in modo da uscire dalle zone scure ed entrare nelle zone più chiare (mantenendo la convenzione per cui a zero è associato il colore nero). La magnitudine sarà tanto più grande quanto più grande il contrasto, quindi differenza di colore e luminosità.

Ora possiamo procedere con la descrizione del filtro di Sobel. Il *Sobel filter* viene utilizzato per creare immagini in cui si enfatizzano gli *edge*. Per *edge* si intendono tutte quelle zone dell'immagine che corrispondono a bordi, margini o spigoli degli oggetti rappresentati nell'immagine. Ciascuna di queste zone deve necessariamente essere associata almeno ad un cambio di colore oppure ad un cambio di luminosità, altrimenti l'oggetto in questione non sarebbe distinguibile dallo sfondo. Quindi, osservando il gradiente dell'immagine, ad ogni *edge* corrisponderà una serie di vettori con magnitudine più grande rispetto alle magnitudini dei vettori circostanti. Lo scopo del *Sobel operator* è generare una approssimazione del gradiente dell'immagine sfruttando due convoluzioni distinte con uno specifico kernel. Nonostante il risultato sia abbastanza grossolano, la sua efficacia e rapidità lo rendono uno dei principali strumenti per la *edge detection*, tecnica che esploreremo fra poco.

L'applicazione del filtro Sobel avviene come mostrato nelle due equazioni sottostanti, dove $*$ denota una convoluzione ed I è un'immagine in scala di grigi.

$$G_x = I * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.4}$$

$$G_y = I * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.5}$$

$$G = \sqrt{G^2 + G^2} \tag{2.6}$$

La prima equazione fornisce un'approssimazione del gradiente rispetto all'asse x, mentre la seconda rispetto ad y. La terza equazione, invece, combina le precedenti fornendoci informazione riguardo alla magnitudine del gradiente dell'immagine.

Ora verranno fatte delle considerazioni sul primo kernel ma, poiché il secondo è una semplice trasposta del primo, tali considerazioni sono, facendo riferimento all'asse delle y, valide anche per il filtro in equazione 2.5. Il kernel nell'equazione 2.4 assegnerà valori in assoluto più grandi a pixel in una posizione di transizione di colore, rispetto a pixel posizionati in una zona con colorazione uniforme. Prendiamo un pixel p di colore bianco posizionato in una zona in cui tutti i pixel sono di colore bianco, si avrà che il kernel, pesando i pixel a destra allo stesso modo di quelli di sinistra ma con segno opposto, attribuisce un valore pari a 0 a p. Se p fosse stato in una zona si transizione dal bianco al nero, avrebbe ottenuto un valore negativo molto grande: tutti i valori nella prima colonna del filtro verrebbero moltiplicati per 255 mentre tutti quelli dell'ultima colonna verrebbero annullati, essendo moltiplicati per 0.

Nell'immagine sottostante viene mostrata l'applicazione dell'operatore Sobel prima rispetto ad x, poi rispetto ad y ed infine il risultato della combinazione delle precedenti.

Figura 2.7: TODO caption

2.2.9 Canny Edge Detection

2.2.10 Circle Hough Transform

Descrizione radius-and-angle parametrization
Descrizione Hough Line Transform
Descrizione Hough Parameter Space
Descrizione algoritmo di Hough Circles

2.2.11 Applicazione degli algoritmi descritti

2.3 Data Augmentation

Due metodi principali: - rotazione - generazione degli scarti sintetici

2.3.1 Generazione Scarti Sintetici

Scarti Sintetici I ritagli non sono stati semplicemente incollati sui conformi: la luminosità della colla è stata modificata per avvicinarsi a quella del pezzo conforme; dopo aver aggiunto la colla è stato praticato uno smooth lungo il contorno, per evitare che ci fosse una transizione netta fra sofndo ed inizio bordo della colla.

Capitolo 3

Gli Autoencoder

3.1 La Struttura di un Autoencoder

TODO quote from wikipedia
Un Autoencoder (AE) è un particolare tipo di rete neurale il cui scopo è codificare efficacemente l'informazione fornita in input con modalità unsupervised. Gli Autoencoder sono stati largamente utilizzati come tecniche per la dimensionality reduction, dimostrando capacità paragonabili a quelle di algoritmi come PCA o t-sne. Un Autoencoder è formato da due componenti principali:

- l'encoder, ottimizzato per generare una version compressa del dato;
- il decoder, ottimizzato per ricostruire l'informazione originale a partire dalla versione compressa.

Durante l'allenamento (training) la rete cerca di fornire un output il più simile possibile all'input ricevuto. Potrebbe sembrare che l'obiettivo dell'Autoencoder sia simulare la funzione identità, cioè quella funzione che fornisce in output esattamente l'input ricevuto. In realtà all'interno della rete accade molto di più. Prendiamo in esempio il più semplice degli AE: due layer fully-connected messi in sequenza. Sappiamo che il numero di feature [1] in ingresso nel primo layer deve combaciare con il numero di feature in uscita, altrimenti non potremmo confrontare l'input con la ricostruzione generata dalla rete. Se si immaginano le feature in ingresso come dei punti, vettori, in un spazio n-dimensionale, con n pari al numero delle feature fornite, il compito dell'Autoencoder è generare dei punti che siano molto vicini ai punti osservati. In altre parole si vuole che l'Autoencoder catturi la distribuzione probabilistica dei vettori forniti. Stabilito, quindi, che la rete riceve e restituisce vettori di

[1] TODO spiegare cosa si intende per feature

dimensionalità n si può definire la dimensionalità dello spazio latente, cioè di quello spazio di dimensione m, con $m < n$, in cui tutti i vettori in input vengono mappati durante l'encoding. Viene chiamato spazio latente perché non è conosciuto a priori. Infatti sarà proprio la rete stessa, durante il training, a trovarlo, selezionandolo tra gli infiniti spazi m-dimensionali. La dimensione dello spazio latente deve essere sufficientemente grande da permettere di mantenere le informazioni che caratterizzano l'input, ed allo stesso tempo sufficientemente piccolo così da rimuovere eventuale rumore o dati superflui. Poco fa abbiamo definito due layer fully-connected, specificando soltanto le dimensioni in input del primo e le dimensioni in output del secondo. Ora sappiamo che le feature in uscita dal primo layer dovranno essere accettate, in entrata, dal secondo. Sappiamo anche che la dimensione della feature, in quanto vettore dello spazio latente, dovrà essere pari ad m. La rete appena definita e raffigurata in con la tipica forma a clessidra. La parte più stretta di un Autoencoder è detta bottleneck (in italiano: collo di bottiglia) e corrisponde con la compressione massima del dato.

3.1.1 Variazioni dell'architettura

Breve descrizione diconvAE stacked SAE DAE VAE

3.1.2 Applicazioni principali

La capacità degli Autoencoder di ricostruire l'input, quindi di mantenerne la struttura, privandolo di eventuale rumore o addirittura rimpiazzando il rumore con valori verosimili, si è rilevata utile in molti campi.

Quando il rumore incide notevolmente

In questo paper si usano gli AE per colorare immagini un bg

Anche in questo documento la rimozione del rumore sarà un tema centrale, l'applicazione nel nostro caso verrà analizzato nel dettaglio nei prossimi capitoli.

3.2 Convoluzioni e Convoluzioni Trasposte

3.3 Spazi Latenti

Capitolo 4

Valutazione Sperimentale

Capitolo 5

Risultati Ottenuti

Capitolo 6

Metodi Alternativi

Capitolo 7

Ulteriori Risultati

Capitolo 8

Conclusioni

Appendice A

Come si fanno le appendici

Le appendici si fanno con `\appendix` seguito da `\chapter{...}`

Appendice B

Esempi di Citazioni Bibliografiche

Pŷrłå in [2] ha poi generalizzato i risultati di Bišker [1].

Il pacchetto `uniudtesi` carica automaticamente `hyperref`, che a sua volta rende "cliccabili" i riferimenti bibliografici nel documento elettronico.

Appendice C

Ambiente GNU/Linux (ad esempio Ubuntu)

Contributo di
Leonardo Taglialegne

Gli ambienti GNU/Linux contengono parecchi strumenti utili per la stesura di una tesi di laurea, in particolare segnaliamo:

- Kile
- KBibTeX

Il primo è un editor per il LaTeX, che include una tabella dei simboli, la visualizzazione della struttura, evidenziazione del codice e simili comodità, e nelle ultime versioni fornisce una visualizzazione in anteprima dei risultati di compilazione.

Il secondo è uno strumento di ricerca, modifica ed inserimento di citazioni in formato BibTeX.

I pacchetti relativi (ed altri utili) si installano, su ambienti Debian e Ubuntu con: `sudo apt-get install kile kile-l10n kbibtex texlive-science texlive-math-extra texlive-lang-italian`

Bibliografia

[1] J. Bišker, *On the elements of the empty set.* Mathematica Absurdica **132** (1999), 13–113.

[2] U. Pŷrłå, *Generalization of Bišker's theorem.* Paperopolis J. Math. **14** (2001), 125–132.