View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Electronic Thesis and Dissertation Archive - Università di Pisa

UNIVERSITÀ DI PISA

Facoltà di Scienze Matematiche Fisiche e Naturali

Corso di Laurea Specialistica in Tecnologie Informatiche

Dipartimento di Informatica

Tesi di Laurea Specialistica

Tecniche di segmentazione di immagini di fondali marini per la mappatura della *Posidonia oceanica*

Laureando

Emanuela Del Dottore

Relatore

Prof.ssa Cecilia Laschi

Prof.ssa Barbara Mazzolai

Dott. Giacomo Santerini

Controrelatore

Prof.ssa Chiara Bodei

Anno Accademico 2009-2010

Ai miei nipotoni

Ringraziamenti

Questa tesi è dedicata a coloro che tanto hanno sperato di giungere a questo giorno, i miei nipoti Mattia e Sebastiano, che ringrazio per tutto l'affetto che mi danno. Con loro anche la mia famiglia, i miei genitori e le mie sorelle, che mi hanno sempre sostenuto.
Per la realizzazione di questa tesi un ringraziamento doveroso va alla Prof.ssa Laschi, a Barbara e Giacomo. Inoltre vorrei ringraziare i ragazzi del laboratorio di Robotica Marina per la loro simpatia e disponibilità.
Grazie infine ai miei preziosi amici.

8 Ottobre 2010

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

La percezione visiva fornisce all'uomo una quantità innumerevole di sensazioni e di conoscenza. Il processo di elaborazione delle informazioni che arrivano dall'apparato visivo è particolarmente complesso, strettamente legato al sistema nervoso e fortemente radicato nella base di conoscenza individuale.

Le deduzioni che l'uomo è in grado di elaborare a partire dalle immagini che arrivano al cervello sono piuttosto articolate, si riesce per esempio a distinguere i diversi materiali senza dover aggiungere informazioni tattili. Si può affermare inoltre che l'uomo è un'entità complessa in grado di apprendere nell'ambito della percezione.

La Computer Vision e il Machine Learning sono le scienze che ambiscono a riprodurre queste capacità su agenti autonomi.

Sono discipline ampie che coinvolgono una vastità di altre materie di studio con innumerevoli applicazioni. Una fra tutte, l'Image Processing.

L'Image Processing è la disciplina che si pone l'obiettivo di estrarre informazioni dalle immagini tramite i processi di analisi e di elaborazione.

Tante sono le tecniche di analisi e di elaborazione di immagini studiate nel corso degli anni e tutt'oggi si ricercano nuovi algoritmi e tecnologie per velocizzare e migliorare il processo di elaborazione.

Ciò che si ottiene dall'elaborazione va a costituire, in ambito del Machine Learning, una base di conoscenza. È perciò un processo delicato ed importante.

Per analisi ed elaborazione delle immagini si intende una serie di metodologie in grado di individuare all'interno di una immagine, gli elementi in

essa racchiusi, quali regioni con particolari caratteristiche, oggetti o primitive più elementari. L'operazione che permette di suddividere una immagine in zone di interesse è detta operazione di segmentazione, mentre l'attribuire alle zone individuate una particolare etichetta che le qualifica per permetterne un riconoscimento in tempi successivi è detta operazione di classificazione.

Il lavoro di tesi si è concentrato sulla ricerca e la creazione di un modello per il riconoscimento di una categoria particolare di oggetto, la *Posidonia oceanica*, a partire da immagini di fondali marini.

Per poter stabilire la presenza o meno di una qualsiasi classe di oggetti all'interno di una immagine occorre, prima di tutto, segmentarla, quindi individuare quali sono le aree ritenute simili, per un qualche criterio, all'interno dell'immagine.

Questa operazione è stata eseguita in questo lavoro di tesi sfruttando la definizione di istogramma come rappresentazione delle distribuzioni delle intensità di una immagine. A partire da un insieme adeguato di filtri, si possono estrarre dagli istogrammi, calcolati sulle risposte ai filtraggi, le strutture sintattiche che vanno a comporre un oggetto contenuto in una immagine.

Si può definire una similarità tra due immagini calcolando la distanza dei relativi istogrammi. In questo lavoro tale distanza è stata misurata con la statistica χ^2. Gli spettri degli istogrammi con le distanze associate risultano una rappresentazione del modello Posidonia.

È stata proposta una selezione di filtri di Gabor per l'estrazione delle strutture sintattiche e le Support Vector Machine come strumento di classificazione.

Gli esperimenti di classificazione hanno portato buoni risultati con tassi di accuratezza del $70 \div 80\%$.

Dopo aver descritto le motivazioni che hanno supportato questa tesi, nel Capitolo 1, e gli obiettivi che si intende raggiungere, si procederà dapprima, nel Capitolo 2, con una breve rassegna sulle tecniche di riconoscimento e di classificazione più diffuse nella comunità scientifica, e poi, nel Capitolo 3, con una descrizione dettagliata sulla costruzione del modello di riconoscimento per la Posidonia, presentandone anche una sua implementazione. Infine nel Capitolo 4 verrà esposto il processo utilizzato per la validazione del modello di riconoscimento e i risultati ottenuti dalla classificazione.

CAPITOLO 1

Motivazioni e Obiettivi

Già dalla metà del secolo scorso numerose e sempre più incalzanti sono state le campagne per la salvaguardia dell'ambiente. Tematica sempre più sentita anche nel mondo della ricerca tecnologica che, stimolata anche dalle richieste provenienti dal Parlamento Europeo, stá proponendo sempre più progetti rivolti al monitoraggio ed al miglioramento delle condizioni ambientali nelle quali tutti noi viviamo.

Ogni forma di inquinamento, che sia di origine antropica o naturale, innesca pericolosi meccanismi a catena sugli ecosistemi ambientali che ne risultano mutati, con danni permanenti o addirittura estinti. Sicuramente una delle risorse più importanti e sulla quale si stá riversando molta attenzione per la sua universale importanza, è l'acqua.

Uno dei più importanti ecosistemi marini che risulta minacciato dal degrado delle condizioni ambientali è rappresentato dalle praterie di *Posidonia oceanica*.

1.1 Posidonia Oceanica

La *Posidonia oceanica* [1][1] è una pianta marina appartenente alla famiglia delle Posidoniacee, autoctona del Mar Mediterraneo nel quale ha trovato le

[1] http://www.minambiente.it Ministero dell'Ambiente e della tutela del territorio e del mare

condizioni ottimali di temperatura, salinità e trasparenza della acque. Cresce tra 0 e 40 metri di profondità, la variazione del limite inferiore è indotta dalla trasparenza delle acque. Colonizza vaste aree dei fondali mediterranei costituendo delle vere e proprie praterie sommerse (posidonieti) che rappresentano una delle componenti fondamentali dell'equilibrio e della ricchezza ambientale del litorale costiero.

La pianta è costituita da un fusto, chiamato rizoma, che è in grado di svilupparsi sia orizzontalmente che verticalmente consentendole di adattarsi alle diverse condizioni ambientali, quali luce o sedimentazione. Predilige fondali mobili (sabbiosi o fangosi) ma si può trovare anche in zone rocciose.

Dal rizoma si sviluppa da un lato l'apparato radicale e dall'altro ciuffi di 6 o 7 foglie nastriformi larghe circa 1 cm e possono arrivare fino a 1 m di lunghezza.

Le praterie di Posidonia rappresentano una delle maggiori fonti nutrizionali nelle zone del Mediterraneo, garantiscono infatti una produzione di circa 38 tonnellate annue di biomassa e producono giornalmente da 10 a 15 litri di ossigeno. Rappresentano uno tra i più estesi e produttivi ecosistemi nella regione mediterranea, fornendo anche ospitalità a una elevata varietà di specie animali e vegetali che trovano rifugio e vasta disponibilità di cibo. Si possono trovare infatti dalle forme più semplici come spugne e celenterati a forme più complesse come crostacei, molluschi e addirittura vertebrati come i pesci, tra cui molti di importanza commerciale.

Oltre che produttrici di cibo e rifugio, le estensioni di praterie di Posidonia risultano utili per il consolidamento del fondale marino e proteggono le coste dai fenomeni di erosione. Infatti la Posidonia ha bisogno per crescere, di una lunga catena evolutiva che prepara il substrato: prima dell'insediamento della Posidonia numerose altre specie di alghe marine devono radicarsi sul fondale, portando ad un innalzamento dello stesso e fornendo così delle barriere naturali per la protezione della costa. Inoltre le foglie della Posidonia agiscono da fattore di attrito sul moto ondoso che viene quindi rallentato addolcendo l'impatto sui litorali. È stato stimato [1] che la regressione di un metro di prateria comporta una perdita di $15 \div 18$ metri di litorale sabbioso.

Le maggiori minacce all'estensione delle praterie risultano essere:

- opere rigide costiere, quali porti, pontili, difese artificiali, etc., spesso effettuate senza il rispetto dei limiti e con uso di materiali non idonei all'incontaminazione delle acque;

- scarichi urbani ed industriali;
- pesca a strascico sotto costa condotta illegalmente;
- ancoraggi incontrollati;
- impianti di maricoltura intensiva che rilasciano mangimi e sostanze medicinali, e ombreggiano il fondale;
- rapida ed incontrollata diffusione di specie alloctone aggressive che vanno sostituendosi alla Posidonia, una delle minacce più grandi per la Posidonia viene dalle specie del genere Caulerpa che hanno una capacità riproduttiva più rapida ed efficace della Posidonia e tendono ad insediarsi in qualsiasi tipo di fondale, anche roccioso e con qualsiasi condizione di luminosità, limpidità ed inquinamento.

La perdita o la regressione della Posidonia porterebbe un fortissimo impatto sulla rete trofica innescando una reazione a catena con molteplici e gravi effetti, i principali risultati possono essere:

- riduzione degli habitat con conseguente perdita di biodiversità;
- riduzione della produttività con conseguente danno alla pesca;
- riduzione della funzionalità ecologica, della capacità di trasformazione e metabolizzazione dei carichi trofici con conseguente impatto sulla capacità di risposta all'inquinamento;
- riduzione del controllo sui meccanismi di erosione costiera con conseguente perdita delle spiagge e danneggiamento alle attività produttive;
- riduzione della qualità ambientale e del valore patrimoniale, naturalistico, scientifico e turistico.

Nonostante la sua rilevante importanza all'interno dell'intero ecosistema e i numerosi interventi fatti per la sua salvaguardia, la regressione delle praterie di *Posidonia oceanica* nel corso dell'ultimo secolo si avverte ed è senza dubbio preoccupante. Nel corso degli anni sono state effettuate numerose indagini per monitorare lo stato di salute e la distribuzione della *Posidonia oceanica*, ciò nonostante sembra tutt'oggi mancare un quadro esauriente della sua distribuzione nelle zone del Mediterraneo.

1.2 Obiettivi del lavoro di Tesi

L'obiettivo di questa tesi è quello di fornire uno strumento di analisi di immagini di fondali marini per il riconoscimento al loro interno della presenza della *Posidonia oceanica*.

La formulazione di un modello di riconoscimento per la Posidonia consentirebbe lo sviluppo di un sistema più complesso preposto all'attività di monitoraggio sulle praterie con il principale obiettivo di automatizzare l'attività di mappatura.

Anche se si sta informatizzando l'attività di raccolta dei dati, la tecnica a tutt'oggi ritenuta più affidabile per la mappatura della *Posidonia oceanica* rimane l'immersione di personale scientifico subacqueo. Sono state comunque proposte strategie alternative nel corso degli anni e spesso adottate tecniche di monitoraggio miste; una delle metodologie più quotate prevede, per l'acquisizione dei dati, l'impiego di scandagli sonori, ma tale sistema ha la pecca di risentire di echi di disturbo e del moto ondoso, inoltre il segnale può risultare alterato su fondali rocciosi e ha difficoltà ad operare su profondità elevate o fondali ad alta pendenza.

L'idea che ha spinto questa tesi è stata quella di poter integrare a basso costo in un sistema autonomo già esistente una funzionalità aggiuntiva quale il monitoraggio dei fondali.

Infatti, presso il "Centro di Ricerca nel campo delle Tecnologie per il mare e della Robotica Marina" della Scuola Superiore Sant'Anna di Pisa è in fase di realizzazione HydroBot, un catamarano in grado di navigare autonomamente per l'adempimento delle missioni che gli vengono richieste.

HydroBot viene realizzato all'interno di un progetto, HydroNet[2][2], che prevede la collaborazione a livello europeo tra enti pubblici, università, centri di ricerca e industrie, finalizzato alla progettazione e realizzazione di una rete di sensori e robot acquatici autonomi integrati in una infrastruttura informatica di supervisione, rivolti al monitoraggio delle acque costiere, lagunari e fluviali.

La rete di sensori sarà in grado di campionare ed analizzare diversi parametri chimico-fisici dell'acqua per cercare in tempo reale la presenza di inquinanti e mantenere un costante controllo sullo stato di salute delle ac-

[2]http://www.hydronet-project.eu/index.php?menu=objectives

que. Il robot autonomo, oltre agli strumenti per le analisi chimico-fisiche, è equipaggiato da un rilevatore GPS e da un sensore di profondità.

HydroBot sarebbe quindi una piattaforma ideale sulla quale collocare una fotocamera per la realizzazione di un sistema di visione per il monitoraggio dei fondali. Un sistema di visione, in grado di acquisire ed elaborare informazioni sui fondali marini, diventerebbe un mezzo per accrescere le conoscenze riguardo all'ambiente, fornendo informazioni di tipo diverso rispetto alle analisi chimico-fisiche; si arriva a controllare lo stato delle acque, nonché le condizioni di inquinamento ambientale, tramite il monitoraggio della flora sottomarina. L'inquinamento produce variazioni nei ritmi biologici, perciò monitorando le variazioni nel tempo e nello spazio dei sistemi biologici marini si possono dedurre le condizioni di salute delle acque e dell'ambiente.

Da qui è sorta la necessità di verificare la possibilità di formulare una tecnica di segmentazione di immagini sottomarine, nello specifico caso applicata al riconoscimento della *Posidonia oceanica*.

Verrà quindi presentata in questa tesi la formulazione del modello di riconoscimento proposto e una sua implementazione, fornendone i risultati che ne convalidano l'efficacia; con l'ambizione di fornire un efficace strumento di segmentazione da poter offrire come punto di partenza per operare anche su scenari diversi.

CAPITOLO 2

Stato dell'Arte

Nel presente capitolo verranno discusse le tecniche e gli approcci più comunemente diffusi nella comunità scientifica, per la risoluzione dei problemi di Pattern Recognition e Classificazione.

La funzione principale di questo capitolo è quella di raccolta e riorganizzazione delle metodologie e degli strumenti usati in queste aree di ricerca. Le nozioni fornite non saranno del tutto esaustive né copriranno la totalità delle possibili argomentazioni proprie dell'analisi ed elaborazione delle immagini, ma sono esposte con il semplice scopo di fornire una base di conoscenza e un appropriato dizionario per la comprensione dei problemi e delle scelte prese nel corso del lavoro di tesi.

Il capitolo fornirà prima di tutto una inquadratura sulla Computer Vision e approfondirà la Pattern Recognition nel paragrafo successivo, discutendo separatamente i diversi possibili approcci usati nella risoluzine dei problemi di riconoscimento su immagini. Proseguirà in Sezione 2.3 con la Rappresentazione dei Dati ed esporrà successivamente (in Sezione 2.4) i maggiori tipi di Classificatori diffusi nella comunità del Machine Learning e della Pattern Recognition. Verrà inoltre esposta, in Sezione 2.5, una breve discussione su di un particolare tipo di pattern, le texture, ed infine, in Sezione 2.6, verrà presentato un esempio di integrazione tra le varie discipline e metodologie.

Gli argomenti trattati di seguito sono ampiamente documentati in letteratura, potrebbero perciò essere innumerevoli i riferimenti per le definizioni

fornite, si propone comunque come buona lettura di approfondimento il [3] sul quale si è fatto largo affidamento, a questo si aggiungono integrazioni provenienti da [4] per quanto riguarda la trattazione dei classificatori, inoltre, per una maggiore comprensione delle Support Vector Machine un buon articolo guida è [5].

2.1 Computer Vision

Un'immagine è la rappresentazione di una scena prodotta dalla riflessione di una data energia, proveniente da una sorgente, sugli oggetti appartenenti alla scena. Le sorgenti possono essere di carattere luminoso, raggi X, infrarosso o anche ultrasuoni, per questo le tipologie di immagini sono svariate. Deve esistere un sensore capace di interpretare l'energia emessa fornendo così la rappresentazione dell'immagine. In termini matematici un'immagine è un insieme di valori che una funzione *f(x)* assume in uno spazio d-dimensionale. Per permettere ad un calcolatore di interpretare un'immagine bisogna passare attraverso un processo di digitalizzazione che trasforma le informazioni prodotte dal sensore in una rappresentazione numerica interpretabile matematicamente. Perciò l'immagine digitale ottenuta da una macchina fotografica è vista come una matrice $I \in \mathbb{R}^{m \times n}$ le cui celle sono chiamate pixel e rappresentano l'intensità luminosa rilevata nel punto $(i, j), \forall i, j$ con $i = 1 \dots m, j = 1 \dots n$.

L'analisi ha lo scopo di estrarre dalle immagini informazioni significative e complesse che riguardano gli oggetti, o regioni, presenti nella scena. Queste informazioni costituiscono una descrizione, ad un qualche livello, della scena nell'immagine. Un'analisi dell'immagine a basso livello è orientata all'analisi delle informazioni che possono essere fornite a livello dei pixel, ossia è rivolta al rilevamento dei contorni, o alla segmentazione dell'immagine, alla divisione dell'immagine in regioni definite per un qualche criterio omogenee. Un'analisi dell'immagine ad alto livello invece, coinvolge quella serie di procedure rivolte al riconoscimento degli oggetti nella scena e si fonda su una precedente analisi a basso livello.

Queste sono tutte operazioni che l'apparato visivo umano svolge. Nonché nell'uomo le informazioni rilevate vengono passate al cervello ed elaborate per essere completate e classificate, ricorrendo anche al bagaglio di conoscenza personale dell'individuo. La Computer Vision è la scienza che ha l'ambizio-

ne di riprodurre in modo automatico, sui calcolatori, il processo visivo e di elaborazione come svolto dall'uomo per l'interpretazione della realtà.

La Computer Vision è una disciplina ampia che comprende ed è costituita da molte e vaste altre aree di studio come la Pattern Recognition, l'Image Processing e il Machine Learning; come tanti e vari sono i campi applicativi, per esempio in robotica, in medicina, in neurologia, in biologia, nell'ingegneria industriale e ovviamente in ambito militare. Perciò i problemi da affrontare e risolvere nel campo della Computer Vision sono dei più svariati ma si possono comunque individuare una serie di attività comuni nel processo di analisi e risoluzione dei problemi.

Prima di tutto c'è la fase di acquisizione dei dati, in una delle forme accennate prima deve essere raccolta l'immagine, può essere quindi pre-elaborata in modo da migliorarne la qualità riducendone il rumore o applicando una qualche trasformazione come per esempio una riduzione di scala per renderla più maneggiabile nelle successive fasi. Alla pre-elaborazione segue l'analisi e l'elaborazione dell'immagine, ossia si ricercano quelle che sono le caratteristiche salienti, chiamate *feature*. Sulla base delle feature si può eventualmente segmentare l'immagine, ossia individuare intere regioni o oggetti di interesse rappresentati da un qualche criterio di similarità tra feature. Una volta segmentata l'immagine si può procedere con una elaborazione a più alto livello, mirata alla classificazione delle regioni individuate in vere e proprie classi di appartenenza, secondo modelli di riferimento, di cui se ne possiede una descrizione per ogni classe negli stessi termini delle feature estratte dall'immagine. Un sistema visivo strutturato in queste fasi è in grado di apprendere la conoscenza dell'ambiente circostante che gli serve per prendere delle decisioni circa eventuali attività da svolgere.

Le fasi più delicate dell'intero processo sono essenzialmente l'addestramento del sistema, quindi come viene formata la conoscenza sulla base della quale il sistema prenderà le decisioni, e quali e quante debbano essere le caratteristiche da estrarre; la rappresentazione dell'immagine in termini di feature deve essere infatti completa e soddisfacente per ottenere una buona discriminazione tra le diverse classi che ci interessa individuare ma allo stesso tempo non deve essere una rappresentazione troppo complessa o troppo ampia in modo da non introdurre dati da processare inutilmente, perché non abbastanza discriminanti, o dati troppo legati all'immagine rilevata in un particolare momento, così da avere una rappresentazione poco generalizzabile ad una

intera categoria di oggetti.

La parola *oggetto* potrebbe riportarci alla mente un'entità ben definita ed identificabile magari tramite una precisa forma geometrica, legata alla nostra concezione di oggetto in ambiente 3-dimensionale, ma in realtà vogliamo che il termine oggetto generalizzi un concetto più ampio, ossia un insieme di strutture, che possono anche non essere definibili geometricamente, ma che nel loro insieme costituiscono una qualche entità alla quale si può assegnare un nome specifico a seconda delle caratteristiche distintive, usiamo perciò al posto di *oggetto* la parola *pattern*, più comunemente diffusa in questa area di studio.

Prima però di fornire maggiori conoscenze sulla rappresentazione delle immagini viene data una inquadratura generale della Pattern Recognition, disciplina importante per la Computer Vision. Per il prossimo paragrafo occorre comunque tenere a mente che alla classica interpretazione della parola immagine si può intercambiare la sua alternativa rappresentazione, in termini di feature, che è stata accennata sopra.

2.2 Pattern Recognition

La Pattern Recognition [3] è lo studio di come le macchine possono osservare l'ambiente circostante, apprendere come distinguere i pattern di interesse dallo sfondo, e fare del lavoro o prendere delle decisioni ragionevoli circa le categorie dei pattern.

Una fase molto importante della Pattern Recognition è quindi il riconoscimento, definito anche classificazione, di un dato pattern. Si possono distinguere due grandi tipologie di classificazione: 1. *classificazione supervisionata*, se si conosce fin dall'inizio il dominio delle classi del riconoscimento; 2. *classificazione non supervisionata*, se non si conoscono a priori le possibili classi del riconoscimento ma devono essere dedotte dai dati in ingresso.

Per risolvere i problemi di Pattern Recognition esistono quattro approcci principali: 1. *template matching*, 2. *approccio statistico*, 3. *approccio sintattico* o *strutturale* e 4. *reti neurali*. Questi non sono certamente gli unici, ma sicuramente sono i più comunemente usati e da più tempo diffusi. Non si possono neanche ritenere tra loro strettamente indipendenti e spesso lo stesso problema di Pattern Recognition può essere interpretato e modellato tramite

differenti approcci. Inoltre possono essere usate in cascata anche più tecniche risolutive per arrivare ad ottenere una migliore efficacia.

Perciò le divisioni tra tipologie di classificazione e i vari approcci risolutivi sono puramente convenzionali, utili per fornire una definizione più o meno formale e inquadrare più chiaramente il problema ma di fatto le varie tecniche e metodologie collaborano tra loro.

2.2.1 Template Matching

Uno dei primi e più semplici approcci alla pattern recognition è basato sul template matching. Il *matching* è un'operazione di confronto usata per determinare la similarità tra due entità dello stesso tipo. Il *template* può essere identificato come un prototipo di un pattern che deve essere riconosciuto. Il pattern che deve essere identificato viene quindi confrontato con il prototipo, template, prendendo in considerazione tutte le possibili posizioni, traslazioni e rotazioni, e le diverse possibili dimensioni di scala. Spesso il template stesso viene appreso dal training set. *Training set* si può letteralmente tradurre con "insieme di addestramento", ed è di fatto identificabile con un insieme di pattern che servono per allenare il riconoscimento; quindi fanno parte del training set un insieme più o meno sostanzioso, a seconda delle esigenze, di campioni delle diverse classi di pattern.

Questo approccio opera a basso livello, il confronto è infatti svolto a livello di pixel. Viene fissata una dimensione per una finestra che scorre sull'immagine; all'interno di questa finestra si ricerca il template, ossia si confronta pixel a pixel il prototipo con la finestra sull'immagine campione.

L'operazione di confronto è computazionalmente dispendiosa visto anche il potenzialmente elevato numero di iterazioni, inoltre può anche richiedere un'occupazione di memoria non indifferente per poter immagazzinare uno o anche più prototipi, tuttavia con gli attuali processori e capacità di memoria anche il Template Matching è diventato un approccio competitivo. La pecca più grande deriva invece dalla rigidità imposta dalla presenza di un template per la quale si può verificare un fallimento in caso di distorsione dell'immagine, per un cambio di prospettiva oppure in caso si abbia una elevata varianza tra i valori delle feature dei pattern appartenenti alla stessa classe.

Nel campo dell'Object Recognition questo è un approccio molto usato, anche intuitivamente ci è chiara la semplicità di un confronto tra forme, qua-

lora gli oggetti da ricercare abbiano contorni non particolarmente complessi e contenuti stilizzabili, rispetto all'applicazione di tecniche più elaborate.

Un noto esempio applicativo è la ricerca di volti. Infatti un volto ha una struttura molto semplice (forma ovale), e i suoi componenti interni (naso, occhi, bocca) sono altrettanto ben stilizzabili e collocabili geometricamente all'interno del volto.

2.2.2 Approccio Statistico

Ogni pattern si può dire simile, per un qualche criterio, agli altri pattern appartenenti alla stessa classe. Ciò vuol dire che ci sono delle caratteristiche che distinguono le diverse categorie di pattern.

Gli approcci statistici permettono di andare a stimare le distribuzioni dei dati, per ogni classe di pattern appartenenti al training set, per fornire le regole decisionali che determinano quale è la probabilità che un pattern appartenga ad una classe, proprio sulla base dei valori assunti dalle feature nel pattern dato. Possono procedere massimizzando la probabilità a posteriori che un pattern appartenga ad una classe oppure procedendo inversamente assegnano il pattern alla classe che apporta il minor rischio di errore. Le regole statistiche che sono più comunemente usate per prendere le decisioni sono la regola di Bayes e la stima di massima verosimiglianza.

2.2.3 Approccio Sintattico

Questo approccio analizza i campioni del training set, alla ricerca delle caratteristiche strutturali o sintattiche con le quali distinguere i diversi pattern. Per questo approccio è spesso utile interpretare i pattern in modo gerarchico: un pattern è strutturato con una gerarchia di sottopattern più semplici. Gli elementi base vengono chiamati *primitive* che con una serie di inter-relazioni e regole strutturali, vanno a costituire un pattern più complesso.

Per chiarire il concetto prendiamo ad esempio la struttura sintattica di un linguaggio: le frasi costituiscono i diversi pattern, le primitive sono le lettere dell'alfabeto dello specifico linguaggio, le frasi vengono generate secondo le regole grammaticali, queste sono quindi le nostre regole strutturali che legano gli elementi base.

Per distinguere le classi di pattern occorre quindi definire un dizionario degli elementi base e quali sono le regole sintattiche della specifica classe di

pattern. Questo tipo di approccio è perciò applicabile nei casi in cui sia possibile individuare una struttura morfologica particolare all'interno dei pattern. Un'applicazione si ha nella ricerca dei contorni e nel riconoscimento delle forme, ma soprattutto si distingue l'uso di questo approccio nel riconoscimento delle texture.

2.2.4 Reti Neurali

Una rete neurale[4] è un insieme di elementi, chiamati nodi, interconnessi tra loro all'interno di un grafo pesato. I nodi possono essere visti come neuroni artificiali e gli archi pesati sono le connessioni tra neuroni di output e neuroni di input. Le reti neurali sono in grado di apprendere complesse relazioni non lineari tra input e output, usano procedure di training sequenziali e si adattano ai dati.

La rete più semplice è il *percettrone*, ossia un singolo neurone che a partire da un insieme di dati in ingresso esegue una funzione interna e produce un valore di output; per addestrare un percettrone si modificano iterativamente i pesi degli archi in ingresso fino a che non si ottengono i dati desiderati.

Il singolo percettrone non è però in grado di svolgere operazioni non lineari, come per esempio la semplice operazione di XOR, per questo vengono utilizzate topologie di reti più complesse. Queste organizzano i nodi in più livelli, tipicamente si individua un livello di nodi di input, un livello di nodi di output e un livello nascosto. A questo modello di base si aggiungono varianti ricorsive in cui gli output vengono reindirizzati verso i livelli precedenti.

2.3 Rappresentazione dei Dati

L'immagine acquisita e che deve essere elaborata non è altro che un insieme di dati, misurazioni, la cui rappresentazione può cambiare a seconda delle esigenze interpretative e delle tecniche risolutive adottate. Il concetto di feature è comunque un punto chiave in tutti gli approcci precedentemente discussi. Ridefiniamo perciò questo termine: un insieme di feature è un vettore m-dimensionale, dove m è il numero delle feature estratte, rappresentante la struttura del pattern che ci interessa; come già detto questa rappresentazione influenza fortemente i risultati della classificazione, sono perciò ancora attuali gli studi sui metodi di estrazione di tali caratteristiche.

Occorre precisare quale sia la distinzione tra estrazione e selezione di feature visto che anche in letteratura questi due termini vengono intercambiati. La selezione di feature comprende tutti quegli algoritmi in grado di selezionare il migliore sottoinsieme di feature da un insieme iniziale più ampio, mentre l'estrazione di feature comprende tutti quei metodi rivolti alla creazione di nuove feature a partire da un insieme di partenza, quindi raccoglie tutta una serie di algoritmi che operano trasformazioni e combinazioni sulle feature iniziali.

In generale si può susseguire ad una prima estrazione delle feature dai dati rilevati, una selezione per scartare quelle feature poco discriminanti. Ma si può anche verificare di procedere con una sola delle due fasi, ovviamente tutto dipende dal dominio applicativo e dal training set a disposizione.

2.3.1 Estrazione di Feature

I metodi di estrazione delle feature devono determinare il più appropriato sottospazio di dimensionalità m nello spazio, tipicamente più ampio, delle feature originali di dimensione d, dove quindi $m \leq d$.

Il più noto estrattore lineare di feature è il PCA, Principal Component Analysis, che calcola gli m più grandi autovettori della matrice di covarianza $d \times d$ degli n pattern d-dimensionali. La trasformazione lineare si può definire come $Y = XH$, dove X è la matrice del pattern di cardinalità $n \times d$, Y è la matrice del pattern ottenuto dalla trasformazione di cardinalità $n \times m$, e H è la matrice di trasformazione lineare di cardinalità $d \times m$ le cui colonne sono proprio gli autovettori.

Dato che il PCA usa le feature più espressive, ossia gli autovettori dei più grandi autovalori, questo metodo approssima effettivamente i dati attraverso il sottospazio lineare usando il criterio del *mean square error*.

Il PCA è un metodo di estrazione lineare non supervisionato, ma esistono ovviamente anche altri metodi di estrazione più o meno adatti a seconda del tipo delle distribuzioni probabilistiche dei dati.

Tra le tecniche di estrazione non lineari è doveroso menzionare il Kernel PCA[6][7], tecnica strettamente legata alla precedente PCA e applicata sulle SVM che verranno introdotte in seguito.

L'idea di base del Kernel PCA è quella di mappare prima i dati di input in un diverso spazio delle feature F, tramite una funzione Φ non lineare, quale potrebbe essere un polinomio di grado p, e successivamente eseguire la

trasformazione lineare PCA nello spazio mappato. Normalmente lo spazio F ha un'alta dimensione. Per evitare di computare il mappaggio Φ esplicitamente, Kernel PCA utilizza i kernel di Mercer i quali hanno la proprietà di poter essere decomposti in prodotti vettoriali $K(x, y) = \Phi(x) \cdot \Phi(y)$. Come risultato, lo spazio del kernel ha una metrica ben definita. Kernel di Mercer possono essere per esempio polinomi del p-esimo ordine $(x \cdot y + 1)^p$ e kernel Gaussiani $e^{-\frac{\|x-y\|^2}{2\sigma^2}}$.

Preso X come matrice normalizzata $n \times d$ del pattern con media zero e $\Phi(X)$ la matrice del pattern nello spazio F, la trasformazione lineare PCA nello spazio F trova gli autovettori della matrice di correlazione $\Phi(X)\Phi(X)^T$, la quale è chiamata matrice kernel $K(X, X)$. Nel metodo Kernel PCA i primi m autovettori di $K(X, X)$ sono ottenuti per definire la matrice di trasformazione, E. E ha dimensione $n \times m$ dove m rappresenta il numero desiderato di feature, $m \leq d$. I nuovi pattern x sono mappati tramite $K(x, X)E$, e sono ora rappresentazioni generali del training set e non strettamente legate ai valori propri assunti dalle feature.

2.3.2 Selezione di Feature

Dato un insieme di d feature, il problema di selezione delle feature è il problema di determinare il sottoinsieme di dimensione m che induce il più basso errore di classificazione. La necessità di sviluppare tecniche di selezione deriva dalla possibilità di avere un insieme molto ampio di feature, indotto magari dalla fusione di dati provenienti da più sensori oppure dalla integrazione di modelli provenienti da approcci diversi dove i parametri dei modelli compongono le feature.

Preso Y come insieme di feature di cardinalità d e con m numero di feature nel sottoinsieme selezionato X, $X \subseteq Y$, rappresentiamo la funzione di selezione con *J(X)* per l'insieme X. Assumiamo che un valore alto di *J(X)* rappresenta una migliore scelta del sottoinsieme; la scelta più naturale come funzione di selezione è $J = (1 - P_e)$ dove P_e rappresenta l'errore di classificazione. Questo criterio comunque produce una selezione dipendente dallo specifico classificatore utilizzato e dal numero degli elementi usati per l'allenamento e per i test.

Il più diretto approccio alla soluzione del problema di selezione comporterebbe l'esame di tutti gli $\binom{d}{m}$ possibili sottoinsiemi di dimensione m e la

selezione del sottoinsieme con il più alto valore di *J(.)*. Questo tipo di ricerca è impraticabile persino per piccoli valori di m e d.

Un approccio che evita di esaminare in modo esaustivo tutti i possibili sottoinsiemi si fonda sul principio divide et impera; il punto chiave per questo criterio è la proprietà di monoticità della funzione *J(.)* per la quale, dati due sottoinsiemi X_1 e X_2 delle feature, se $X_1 \subset X_2$ allora $J(X_1) < J(X_2)$.

Da questa proprietà deriva che la performance del sottoinsieme migliora quando una feature viene aggiunta al sottoinsieme. In pratica però le funzioni più usate non soddisfano questa proprietà. Le tecniche utilizzate sono quelle computazionalmente più fattibili ma non garantiscono di raggiungere l'insieme ottimale. Per esempio si può decidere di mettere insieme quelle feature che prese singolarmente forniscono una buona discriminazione ma non è detto che messe insieme forniscano una rilevante diminuzione dell'errore. Un'altra tecnica può essere quella di aggiungere (o togliere) sequenzialmente una feature fin tanto che non si ha una diminuzione (aumento) dell'errore troppo piccola (troppo grande).

2.4 Classificazione

La classificazione nella sfera dell'Object Recognition rappresenta quella procedura che permette di associare un oggetto ad una o più etichette rappresentanti le possibili categorie alle quali può appartenere l'oggetto in questione. Quando sono note le possibili classi di appartenenza dei pattern si parla di classificatori supervisionati altrimenti, se le classi devono essere dedotte dai dati a disposizione senza alcuna conoscenza aggiuntiva, si parla di classificatori non supervisionati. Una volta determinato quale rappresentazione utilizzare per i pattern si può modellare un classificatore, che sfrutti un apprendimento supervisionato oppure no, usando uno o più degli approcci già discussi precedentemente nelle Sezioni 2.2.1, 2.2.2, 2.2.3 e 2.2.4.

Prima di procedere con una breve rassegna dei più noti classificatori, occorre riportare alla mente i tre concetti fondamentali che concorrono alla progettazione dei classificatori e che sono già stati incontrati in Sezione 2.2.

Il concetto più semplice e intuitivo che viene sfruttato è quello della *similitudine*: pattern che sono simili devono essere assegnati alla stessa classe. Occorre determinare quale è la metrica di paragone per stabilire la similitu-

dine. Questa ovviamente varia tra un classificatore ed un altro perché varia la rappresentazione stessa dei dati.

Il secondo concetto importante è quello di *distribuzione probabilistica*. Infatti i classificatori più diffusi sono proprio quelli di tipo statistico, o perlomeno la progettazione dei classificatori si appoggia di frequente allo studio delle distribuzioni probabilistiche dei dati per determinare il migliore classificatore.

Il terzo concetto è quello di confine, o *regole decisionali*. Per stabilire la classe di appartenenza occorre fissare le regole con le quali è possibile prendere le decisioni, per esempio occorre determinare quale metrica utilizzare per determinare la similitudine ad un pattern, oppure quale regola statistica applicare o anche come calcolare l'errore fatto dalla classificazione e con questo, quale regola di minimizzazione adottare. Un esempio, quello più diffuso, di criterio di minimizzazione dell'errore è il *mean square error* (MSE), che minimizza la distanza tra l'output del classificatore e un qualche valore obiettivo.

2.4.1 I Classificatori

Abbiamo già enunciato precedentemente le Reti Neurali, che risultano punto di forza per lo sviluppo di sistemi intelligenti e bioispirati, e il Template Matching, classificatore semplice e intuitivo molto apprezzato nell'Object Recognition, ma ce ne sono di molti altri come per esempio il Minimum Distance Classifier, che è una variante del Template Matching, ma a differenza di questo usa più di un prototipo, anche se in numero limitato, per ciascuna classe.

Il Nearest Mean Classifier, noto anche come 1-NN (*one nearest neighbor*), è uno dei più utilizzati e viene spesso usato come termine di paragone per misurare le performance di altri classificatori in un determinato dominio. Questo classificatore si è spesso dimostrato il classificatore con le performance più ragionevoli indipendentemente dal dominio applicativo e dalle implementazioni. Nel 1-NN, ciascuna classe di pattern è rappresentata da un singolo prototipo stabilito come media dei pattern del training set per quella classe. Non richiede nessun parametro, eccetto la distanza da usare come metrica e per convenzione viene utilizzata la distanza Euclidea.

Il k-NN è una variante del 1-NN che fornisce, per il pattern in esame, k possibili classi e sceglie quella con il maggior numero di voti.

Un particolare tipo di classificatore è l'albero decisionale che viene allenato iterativamente selezionando ad ogni passo le feature che sono ritenute più importanti per ogni diverso nodo dell'albero. Perciò ad ogni iterazione non vengono esaminate tutte le feature ma solo il sottoinsieme ritenuto importante.

Classificatori Bayesiani

Abbiamo dichiarato l'approccio statistico essere il più diffuso, è perciò doveroso enunciare i classificatori Bayesiani che, come si può dedurre dal nome, fanno uso del teorema di Bayes per calcolare le probabilità a posteriori, fornendo una stima dei parametri statistici a partire dai dati del training set. Immaginiamo di avere un insieme C di diverse possibili classi, e una serie di d feature $A_1, A_2, \ldots, A_d$. Si può stabilire che un dato pattern appartiene alla classe $c \in C$ se la:

$$\Pr(C = c | A_1 = a_1 \wedge A_2 = a_2 \wedge \cdots \wedge A_d = a_d)$$

è massima. Dal teorema di Bayes questa probabilità è uguale a:

$$max \frac{\Pr(A_1 = a_1 \wedge A_2 = a_2 \wedge \cdots \wedge A_d = a_d | C = c)}{\Pr(A_1 = a_1 \wedge A_2 = a_2 \wedge \cdots \wedge A_d = a_d)} \Pr(C = c) \qquad (2.1)$$

dove C indica la variabile aleatoria delle possibili classi, mentre $a_1, a_2, \ldots, a_d$ sono i valori assunti dalle feature.

La probabilità a priori di avere la classe c è stimabile dal training set, il denominatore del primo termine nella formula (2.1) è lo stesso per tutte le possibili classi, ciò che è difficile determinare è il numeratore della (2.1) perché le feature tra loro non sono indipendenti.

Una versione naive dei classificatori Bayesiani fa un'assunzione molto forte proprio sull'indipendenza delle feature, che siano cioè completamente indipendenti. Ciò consente di fare equivalere il numeratore della (2.1) a:

$$\Pr(A_1 = a_1 | C = c) \Pr(A_2 = a_2 | C = c) \ldots \Pr(A_d = a_d | C = c) \qquad (2.2)$$

A questo punto anche la (2.2) è facilmente ricavabile dal training set.

Nella realtà però non è possibile ritenere le feature indipendenti tra loro né tanto meno che tutti i dati abbiano lo stesso tipo di distribuzione, è per questo che negli approcci statistici viene fatto largo uso delle *misture*.

Le misture sono uno strumento di modellazione flessibile e potente. Nell'approccio statistico il principale uso delle misture si ha nella definizione formale dei classificatori non supervisionati, ma sono anche uno strumento di rappresentazione utile anche in classificatori supervisionati per dati con densità complesse. Sono infatti in grado di modellare situazioni in cui i pattern sono prodotti da un gruppo di modelli (distribuzioni probabilistiche), tra un insieme composto da gruppi di più modelli probabilistici. Vengono inoltre anche usate come strumento di selezione di feature.

Possiamo chiarire il concetto considerando di avere n campionature, $\mathbf{y} = \mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^n$, e di prenderle come modello. Se consideriamo K possibili classi, ciascuna di esse è caratterizzata da una funzione di probabilità $\Pr_m(\mathbf{y}|\theta_m)$ dove $m = 1, \ldots, K$. Ciascun campione può essere associato ad una delle classi con probabilità $\alpha_1, \alpha_2, \ldots, \alpha_K$. Perciò la funzione di probabilità per la variabile $\mathbf{y}$ risulta:

$$\Pr(\mathbf{y}|\Theta_{(K)}) = \sum_{m=1}^{K} \alpha_m \Pr_m(\mathbf{y}|\theta_m)$$

dove $\Theta_{(K)} = \{\theta_1, \ldots, \theta_K, \alpha_1, \ldots, \alpha_{K-1}\}$; ricavare il modello in questo caso vuol dire determinare il valore dei parametri in $\Theta_{(K)}$ che meglio descrivono i dati osservati.

Support Vector Machine

Uno dei più interessanti classificatori introdotti recentemente si basa sul concetto dei vettori di supporto da cui prende il nome, Support Vector Machine (SVM). Questa tecnica viene sviluppata da Vapnik[8],[9] nella seconda metà degli anni 90. Nato come classificatore per singola classe, è successivamente stato adattato anche per il riconoscimento di più categorie.

Data l'introduzione delle SVM la pattern recognition è dovuta ricorrere anche all'Ottimizzazione Combinatoria allargando i suoi studi verso le tecniche di programmazione lineare e programmazione lineare intera.

Infatti il problema della Pattern Recognition viene, in ambito delle SVM, interpretato come problema di calcolo combinatorio.

Seguendo le linee guida del [5], definiamo prima il problema per il caso più semplice in modo da chiarire l'idea che c'è dietro a questo tipo di classificatore,

poi passiamo alla sua formulazione Lagrangiana per spiegare l'approccio usato per la risoluzione dei problemi di Pattern Recognition anche per situazioni non lineari.

Partendo dal caso lineare con dati facilmente separabili, etichettiamo i dati del training set con $\{\mathbf{x}_i, y_i\}, i = 1, \ldots, l, y_i \in [-1, 1], \mathbf{x}_i \in \mathbb{R}^d$. Supponiamo di avere l'iperpiano di separazione tra i valori negativi e i valori positivi. I punti $\mathbf{x}$ che poggiano sull'iperpiano soddisfano i vincoli $\mathbf{w} \cdot \mathbf{x} + b = 0$, dove $\mathbf{w}$ è il vettore normale all'iperpiano. $|b| / \|\mathbf{w}\|$ è la distanza, perpendicolare, tra l'iperpiano e l'origine, e $\|\mathbf{w}\|$ è la norma Euclidea di $\mathbf{w}$. Consideriamo $d_+(d_-)$ come la distanza più corta tra l'iperpiano di separazione e il più vicino valore positivo (negativo). Allora definiamo $d_+ + d_-$ come la *frontiera* dell'iperpiano di separazione. Questo si rivela essere un problema di programmazione lineare, dove viene soddisfatta la proprietà di convessità vista la natura continua dei vincoli, e viene risolto trovando il valore ottimo che si trova proprio sulla frontiera dell'iperpiano di separazione.

Possiamo esprimere il problema in termini di Programmazione Lineare:

$$\min \|\mathbf{w}\|^2 \tag{2.3}$$

$$y_i (\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \forall i \tag{2.4}$$

$$y_i \in [-1, 1] \forall i$$

Il vincolo (2.4) esprime in se due diversi vincoli:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq 1 \text{ per } y_i = 1 \tag{2.5}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \text{ per } y_i = -1 \tag{2.6}$$

Possiamo avere una idea più chiara osservando la Fig. 2.1. I punti per i quali si verifica l'eguaglianza della disequazione (2.5) giacciono sulla frontiera dell'iperpiano $H_1 : \mathbf{x}_i \cdot \mathbf{w} + b = 1$ con normale $\mathbf{w}$ e distanza dall'origine $|1 - b| / \|\mathbf{w}\|$. Analogamente, i punti per i quali si verifica l'eguaglianza della (2.6) giacciono sulla frontiera dell'iperpiano $H_2 : \mathbf{x}_i \cdot \mathbf{w} + b = -1$ che ha

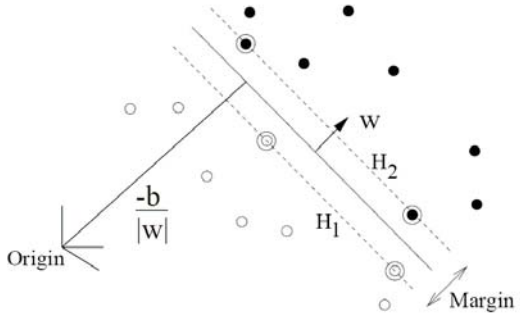


Figura 2.1: Iperpiani di separazione per il caso lineare.

sempre normale $\mathbf{w}$ e distanza dall'origine $|-1-b| / \|\mathbf{w}\|$. Si ottiene da qui che $d_+ = d_- = 1/\|\mathbf{w}\|$, da cui il margine $2/\|\mathbf{w}\|$. H_1 e H_2 sono due iperpiani paralleli e nessun punto ricade tra loro (all'interno del margine). Da qui si ottiene la funzione obiettivo (2.3).

I *vettori di supporto* sono proprio quei punti che ricadono sulle frontiere di H_1 e H_2 e che quindi soddisfano in eguaglianza i vincoli (2.5) e (2.6) (sono i punti marcati con un cerchio in Fig. 2.1).

È utile interpretare questo stesso problema in un'altra forma, ossia con la formulazione Lagrangiana. Introduciamo quindi i moltiplicatori Lagrangiani $\alpha_i, i = 1, \ldots, l$, uno per ogni vincolo in (2.4); per definizione i moltiplicatori Lagrangiani sono tutti positivi. Il problema Lagrangiano cerca di minimizzare la funzione:

$$\frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b\right) + \sum_{i=1}^{l} \alpha_i \tag{2.7}$$

Del problema (2.7) possiamo formulare il relativo duale:

$$\max \left\{ \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j : 0 \leq \alpha_i \leq C, \sum_i \alpha_i y_i = 0 \right\} \tag{2.8}$$

Dove C indica la massima penalità assegnata all'errore, limite superiore per i moltiplicatori lagrangiani.

La formulazione del problema in Duale Lagrangiano (2.8), è utile per generalizzare la procedura del classificatore SVM ai casi non lineari, infatti questa è una forma più facile da gestire all'interno della quale i punti appaiono sotto forma di prodotti vettoriali, $\mathbf{x}_i \cdot \mathbf{x}_j$. Questa particolare proprietà viene sfruttata dal Kernel PCA, tecnica di estrazione di feature già discussa in Sezione 2.3.1, e che rappresenta esattamente l'approccio usato per la risoluzione dei problemi non lineari dal classificatore SVM.

Infatti per il caso non lineare, l'SVM mappa prima i dati in un altro spazio Euclideo H tramite una funzione non lineare $\Phi : \mathbb{R}^d \mapsto H$. A questo punto i prodotti vettoriali dei punti sono ora espressi nella forma $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Se esiste una funzione kernel K tale che $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, non ci sarà più bisogno di conoscere quale sia la funzione non lineare Φ e basterà usare all'interno dell'algoritmo il kernel K.

Anche se siamo nel caso non lineare, con il piccolo trucco di andare a mappare i punti su un diverso spazio Euclideo (H), riusciamo a risolvere linearmente il problema. Il classificatore SVM trova la soluzione del problema calcolando il segno della funzione:

$$f(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{s}_i, \mathbf{x}) \tag{2.9}$$

Dove $\mathbf{s}_i$ sono i vettori di supporto.

2.4.2 Clustering

Il Clustering è un processo di raggruppamento dei dati secondo un qualche criterio di somiglianza ricavato dall'analisi dei dati stessi, senza un apporto di informazione aggiuntiva come l'etichettatura dei campioni del training set nei classificatori supervisionati. Raggruppa in sé tecniche che possono essere catalogate come classificatori non supervisionati.

Gli algoritmi di clustering sono in grado di gestire una grande quantità di dati, sono perciò tipi di classificatori privilegiati in settori come il data mining o l'information retrieval; vengono di norma utilizzati per la segmentazione di immagini, nella compressione e codifica dei segnali e nel machine learning.

Gli algoritmi di clustering si basano essenzialmente su due tecniche: 1. *partizionamento*, i dati vengono partizionati secondo algoritmi euristici ite-

rativi con lo scopo di minimizzare una certa funzione obiettivo; 2. *aggregamento gerarchico*, i dati vengono organizzati in sequenze annidate di gruppi, come le strutture ad albero, sui quali si può operare in modalità *bottom-up* o *top-down* a seconda che si cerchi di aggregare gli elementi per fonderli in un cluster, primo caso, oppure si cerchi di dividere un unico cluster in più possibili sottocluster, secondo caso.

I cluster di tipo partizionale sono quelli usati più di frequente e il problema di clusterizzare partizionalmente può essere interpretato come: dati n pattern in uno spazio d-dimensionale, determinare una partizione dei pattern in K cluster, tali che i pattern in un cluster siano più simili tra loro che rispetto ai pattern appartenenti a cluster diversi. Il valore di K può non essere specificato. Si possono adottare criteri di analisi locale o globale. Un operatore locale forma i cluster usando informazioni locali nei dati, mentre gli operatori globali usano informazioni globali dei dati ossia rappresentano un cluster con un prototipo e assegnano ciascun pattern al cluster con il prototipo più simile. Gli algoritmi di clustering più usati di tipo partizionale sono il K-means e l'Expectation Maximization.

K-means

Cerca di ottenere il partizionamento dei dati minimizzando il valore della somma dei quadrati delle distanze di ciascun pattern con il prototipo più vicino (criterio del *mean square error*). Genera K cluster $\{C_1, C_2, \ldots, C_K\}$ tali che il pattern n_i appartiene esattamente ad un unico cluster e quel cluster sia C_i. Per ogni cluster è definito il centro, o vettore medio come

$$m^{(i)} = \frac{1}{n_i} \sum_{j=1}^{n_i} x_j^{(i)}$$

dove $x_j^{(K)}$ è il j-esimo pattern all'interno del cluster C_i.

La variazione intra-classe tra i pattern dello stesso cluster è definita dal quadrato della distanza Euclidea tra i vari pattern nella classe C_i con il proprio centroide $m^{(i)}$, ossia:

$$e_i^2 = \sum_{j=1}^{n_i} \left(x_j^{(i)} - m^{(i)}\right)^T \left(x_j^{(i)} - m^{(i)}\right)$$

La funzione obiettivo da minimizzare nel metodo di clusterizzazione contenente K cluster, fissato K, è data dalla sommatoria delle distanze intra-classe:

$$E_K^2 = \sum_{i=1}^{K} e_i^2$$

L'algoritmo si può riassumere in quattro passi:

1. seleziona una partizione iniziale di K cluster e se ne fissano i centri
2. assegna ogni pattern al cluster con centro più vicino secondo la minima distanza intra-classe (di solito distanza Euclidea)
3. ricalcola i centri dei cluster come la media tenendo conto dei pattern aggiunti nel precedente passo
4. ripete iterativamente le istruzioni 2 e 3 fino ad ottenere il valore ottimo di terminazione ($\min E_K^2$)

Expectation Maximization

Questo algoritmo si basa sul concetto delle misture di Gaussiane. Ritornando quindi ai concetti introdotti nel paragrafo dei Classificatori Bayesiani, diciamo che l'algoritmo interpreta le campionature $\mathbf{y}$ come dati incompleti, dove i dati mancanti sono l'insieme delle etichette associate a $\mathbf{y}$, $\mathbf{z} = \left\{\mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^n\right\}$. Queste variabili mancanti, $\mathbf{z}^i = [z_1^i, z_2^i, \ldots, z_K^i]$, indicano quale delle K componenti ha generato $\mathbf{y}^i$; se fosse stata l'm-esima componente allora $z_m^i = 1$ e $z_j^i = 0\ \forall j \neq m$.

Avendo ora sia $\mathbf{y}$ che $\mathbf{z}$ riformuliamo la funzione di probabilità come il logaritmo della stima di massima verosimiglianza (*log-likelihood*):

$$L_\ell\left(\Theta_{(K)}, \mathbf{y}, \mathbf{z}\right) = \sum_{j=1}^{n}\sum_{m=1}^{K} z_m^j \log\left[\alpha_m p_m(\mathbf{y}^j|\theta_m)\right] \text{ con } \sum_{i=1}^{K}\alpha_i = 1.$$

L'algoritmo di EM ripete iterativamente due passi, uno chiamato di Expectation, l'altro di Maximization, fino ad ottenere la convergenza del log-likelihood:

- **E-step**: computa il L_ℓ con le $\mathbf{y}$ e l'attuale (iterazione t-esima) stima del parametro $\tilde{\Theta}_{(K)}^{(t)}$, ciò si riduce a trovare la probabilità condizionale delle variabili mancanti: $w_m^{(i,t)} \equiv \Pr(z_m^i|\tilde{\Theta}_{(K)}^{(t)}, \mathbf{y})$
- **M-step**: aggiorna le stime del parametro $\tilde{\Theta}_{(K)}^{(t+1)}$ massimizzandone il valore tenendo conto dei parametri ottenuti al passo E, calcola cioè $\tilde{\Theta}_{(K)}^{(t+1)} = \arg\max_{\Theta_{(K)}} Q\left(\Theta_{(K)}, \tilde{\Theta}_{(K)}^{(t)}\right)$ e aggiorna $\tilde{\alpha}_m^{(t+1)} = \frac{1}{n}\sum_{i+1}^{n} w_m^{(i,t)}$, $m = 1, \ldots, K-1$, per l'iterazione successiva.

2.5 Analisi di Texture

Un tipo di pattern sul quale il mondo dell'analisi di immagine ha riposto un particolare interesse è la *texture*, o grana. Esempi di texture sono le trame e la granularità dei tessuti o l'aspetto dei materiali. In letteratura si possono trovare, per il termine texture, numerose definizioni; alcuni esempi vengono proposti in [10], da questi possiamo estrarre i più interessanti:

- *"We may regard texture as what constitutes a macroscopic region. Its structure is simply attributed to the repetitive patterns in which elements or primitieves are arranged according to a placement rule."[11]*
- *"A region in an image has a constant texture if a set of local statistics or other local properties of the picture function are constant, slowly varying, or approximately periodic."[12]*
- *"The notion of texture appears to depend upon three ingredients: (i) some local 'order' is repeated over a region which is large in comparison to the order's size, (ii) the order consists in the nonrandom arrangement of elementary parts, and (iii) the parts are roughly uniform entities having approximately the same dimensions everywhere within the textured region."[13]*

Le definizioni di texture variano sia a seconda delle persone che la formulano sia dalle applicazioni per la quale viene presa in considerazione la texture, possiamo comunque evincere dalle definizioni estratte che la texture è un particolare pattern sul quale è possibile individuare una struttura, si può pensare come costituita da elementi base interpretati come sottopattern ripetuti con una funzione periodica lungo l'intera immagine di texture e con solo piccole variazioni.

Si può avere il bisogno di segmentare una immagine sulla base delle differenti texture che contiene, una segmentazione che procede con la ricerca e la seprazione di texture è detta appunto *texture classification*.

La classificazione di texture produce una mappa sull'immagine di input che raggruppa regioni uniformi tra loro.

Molte sono le discipline dove il riconoscimento di texture riveste un ruolo importante, come per esempio in campo industriale nell'automazione del

processo di ispezione dei tessuti, in ambito medico nell'elaborazione delle immagini per la ricerca di tessuti tumorali, nell'elaborazione dei documenti testuali o ancora, nell'elaborazione delle immagini provenienti da satellite per il riconoscimento di zone agricole, acqua o nella ricerca di aree di un qualche interesse geologico.

Tra i primi lavori di approfondimento in materia di texture è doveroso riportare gli studi di Julesz, [14], [15], [16], [17] che per primo si è chiesto come fosse possibile distinguere una coppia di texture caratterizzate dalla stessa luminosità, contrasto e colore. I suoi studi hanno puntato su di un approccio statistico con lo scopo di determinare quali proprietà spaziali si potevano ricavare da immagini di texture in scale di grigio.

Prima di fornire l'idea di Julesz occorre dare le definizioni di statistica del primo e del secondo ordine:

statistica del primo ordine misura la probabilità di osservare un valore di grigio ad una locazione casuale nell'immagine. Questo tipo di statistica può essere computata tramite l'istogramma delle intensità di grigio dell'immagine. In questo caso non esistono dipendenze tra pixel vicini. Il calcolo della media sulle intensità dell'intera immagine è un esempio di statistica del primo ordine.

statistica del secondo ordine è definita come la probabilità di osservare una coppia di valori di grigio come estremi di una certa distanza, casuale, posizionati nell'immagine ad una locazione e orientamento casuali. Queste sono proprietà che legano coppie di valori.

Julesz ha formulato l'idea che due texture non sono, in fase preattentiva, distinguibili se la loro statica del secondo ordine risulta identica. Julesz ha proposto la "teoria dei *texton*"[18] per spiegare come distinguere una coppia di texture. *Texton* sono elementi visuali, come punti e linee, la cui presenza e occorrenza caratterizza la texture.

La teoria dei *texton* è stata ripresa e avvalorata da studi successivi, che ne ricercano una definizione, strutture geometriche e modelli generativi che tengano conto di variazioni di luminosità e posizioni. Un approfondimento si può trovare in [19].

La varietà dei tipi di texture è numerosa e per questo non può essere definito un unico modello e metodo di ricerca, per questo anche i metodi di rappresentazione sono numerosi, ma tutti rivolti alla ricerca ed estrazione di

feature proprie della texture da riconoscere. Per esempio si può ricorrere a *metodi statistici*, che calcolano la distribuzione dei valori dei pixel; oppure possono essere usati *metodi geometrici* che ricorrono proprio alla definizione di texture come pattern costituito da strutture elementari disposte secondo una qualche proprietà geometrica e distribuzione spaziale, in questo caso una delle tecniche di definizione per il modello della texture sono le tassellature di Voronoi; altri metodi *basati sui modelli* cercano di catturare le proprietà essenziali delle texture tramite la definizione che viene data di modello, per esempio molte superfici di carattere naturale hanno caratteristiche che si possono ripetere al loro interno a diverse dimensioni e scalature, in questo caso i frattali sono uno strumento adeguato per la rappresentazione di queste strutture; infine possiamo mensionare i *metodi di elaborazione dei segnali*, che operano nel dominio delle frequenze.

2.5.1 Metodi Statistici

Questi sono i metodi che ricorrono alla statistica del primo e del secondo ordine. Per la statistica del primo ordine[20] si considerano i singoli valori di grigio e la loro distribuzione, ossia l'istogramma. La probabilità $\Pr(i)$ che compaia il valore di grigio i è data dal rapporto tra il numero di pixel con valore i e il numero totale dei pixel nell'immagine.

$$0 \leq \Pr(i) \leq 1 \; \forall i, \sum_{i=1}^{N} = 1$$

Si possono estrarre dall'istogramma alcune caratteristiche utili come i momenti di ordine n, con $n > 0$ intero:

$$m_n = \sum_{i=1}^{N} i^n \cdot \Pr(i)$$

ed i momenti centrali di ordine n:

$$C_n = \sum_{i=1}^{N} (i - m_1)^n \cdot \Pr(i)$$

Se considerassimo l'istogramma e i valori statistici su una intera immagine, non si otterrebbero risultati significativi, ma se riduciamo l'attenzione su finestre piccole, specialmente quando si tratta di immagini naturali, allora le

Tabella 2.1: Feature di tessitura estratte dalle matrici di co-occorrenza.

Texture Feature	Forumla
Energia	$\sum_i \sum_j M_d^2(i,j)$
Entropia	$-\sum_i \sum_j M_d(i,j) \log M_d(i,j)$
Contrasto	$\sum_i \sum_j (i-j)^2 M_d(i,j)$
Omogeneità	$\sum_i \sum_j \frac{M_d(i,j)}{1+\lvert i-j \rvert}$
Fattore di correlazione	$\frac{\sum_i \sum_j (i-\mu_x)(j-\mu_y) M_d(i,j)}{\sigma_x \sigma_y}$

caratteristiche estratte dall'istogramma assumono molta importanza per la distinzione di texture diverse.

Passando alla statistica del secondo ordine, ci interessa determinare la probabilità $\Pr_d(i,j)$ che due punti nell'immagine, distanziati da d elementi, abbiano i e j come valori di grigio. Questi valori di probabilità vengono raccolti nella matrice di co-occorrenza:

$$M_d(i,j) = \begin{bmatrix} \Pr_d(1,1) & \dots & \Pr_d(N,1) \\ \vdots & \vdots & \vdots \\ \Pr_d(1,N) & \dots & \Pr_d(N,N) \end{bmatrix}$$

Si cerca di caratterizzare la struttura di queste matrici, e quindi anche quella della tessitura, estraendo da esse delle proprietà significative, tipicamente quelle riportate in Tabella 2.1 [10].

Una proprietà importante nelle texture è la ripetitività degli elementi base nell'immagine. La funzione di autocorrelazione di una immagine può essere usata per stabilire la regolarità come anche la ruvidità di una texture. La funzione di autocorrelazione è definita come nella forumla (2.10).

$$\rho(x,y) = \frac{\sum_{u=0}^{N} \sum_{v=0}^{N} I(u,v) I(u+x, v+y)}{\sum_{u=0}^{N} \sum_{v=0}^{N} I^2(u,v)} \tag{2.10}$$

Se la texture è ruvida, la funzione di correlazione convergerà a zero lentamente, altrimenti se la texture è liscia convergerà molto rapidamente. Per texture regolari invece, la funzione di correlazione assumerà un aspetto ondulatorio.

2.5.2 Metodi Geometrici

Questo tipo di metodi sono legati alla definizione di texture come superficie composta da elementi o primitive. I metodi di analisi dipendono dalle proprietà geometriche di queste primitive, non cercano di estrarre le proprietà statistiche dagli elementi che compongono la texture, ma cercano le regole di collocazione.

Le tassellature di Voronoi sono state ampiamente proposte per la loro caratteristica di definire relazioni spaziali di vicinanza e per la loro capacità di interpretare le distribuzioni spaziali degli elementi con i poligoni di Voronoi[21].

Considerando una arbitraria coppia di punti P e Q, la bisezione della linea congiungente i due punti è quell'insieme di punti equidistanti da entrambi P e Q, e divide il piano in due metà. Ciascuna metà del piano H_P^Q (H_Q^P) è costituita dall'insieme dei punti più vicini a P (o Q). Per ogni scelta di P si ottiene un insieme di mezzi piani per diverse scelte di Q. L'intersezione di questi mezzi piani definisce un poligono contenente i punti più vicini a P (o Q). Questa regione è chiamata poligono di Voronoi associata al punto.

Due punti sono detti vicini di Voronoi se i rispettivi poligoni di Voronoi condividono un vertice. Molti degli ambienti strutturali manifestano le proprietà geometriche dei vicini di Vornoi, è per questo che possono essere usati i poligoni di Voronoi come feature di texture.

Per poter applicare metodi geometrici su immagini in scale di grigio, occorre prima estrarre le primitive delle immagini. Per fare questo sono sufficienti operazioni di filtraggio sulle immagini, tipicamente Laplacian-of-Gaussian. Da qui selezionare quei pixel che giacciono sui massimi locali e su di essi ricercare le componenti connesse, ciascuna componente connessa definisce una primitiva.

Con le primitive estratte vengono successivamente costruite le tassellature di Voronoi. Su ciascuna cella di Voronoi vengono estratte le feature e le feature simili sono raggruppate per costruire regioni uniformi di texture.

2.5.3 Metodi basati sui modelli

Con questi metodi, l'analisi delle texture sono basate sulla costruzione di una immagine modello, ossia una immagine che sia in grado di descrivere la texture e di sintetizzare le qualità essenziali contenute in essa.

Le catene di Markov sono uno dei più popolari metodi di costruzione dei modelli, per la loro capacità di catturare le informazioni locali contenute in una immagine. Infatti, la proprietà Markoviana si basa sulla località, nello spazio o nel tempo. Le catene di Markov sono un modello matematico per sistemi stocastici i cui stati, discreti o continui, sono governati da una probabilità di transizione. Lo stato corrente nelle catene di Markov dipende solamente dai più recenti stati, per una catena del primo ordine si ha che la probabilità di transizione in uno stato x_t dipende solamente dal precedente stato x_{t-1}:

$$\Pr(x_t|x_{t-1}, x_{t-2}, \ldots, x_0) = \Pr(x_t|x_{t-1})$$

Questi modelli assumo che l'intensità di ciascun pixel dipende dalle intensità dei più vicini pixel.

I modelli di Markov sono adottati in varie applicazioni di elaborazione delle immagini come nella classificazione delle texture [22] [23], nella segmentazione di immagini [24], nella sintesi di texture [25] e nella compressione delle immagini.

2.5.4 Metodi di elaborazione dei segnali

Il cervello umano è il primo ad effettuare un'analisi delle immagini nel dominio delle frequenze. Le texture sono particolarmente predisposte a questo tipo di analisi viste le loro proprietà.

L'applicazione di filtri nel dominio spaziale sono il modo più diretto di catturare queste proprietà. Tipicamente vengono applicati in questo dominio algoritmi di ricerca dei contorni. Gli operatori più usati sono le maschere di Robert $R_1 = \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$, $R_2 = \left(\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right)$ o Laplace $L = \left(\begin{smallmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{smallmatrix}\right)$.

Un altro insieme di filtri si basa sui momenti spaziali. I momenti $(p+q)$ lungo una regione R sono ottenuti dalla formula:

$$m_{pq} = \sum_{(x,y)\in R} x^p y^q I(x,y)$$

Se si prende una regione rettangolare e si calcolano i momenti per ogni pixel della regione e si itera su tutta l'immagine, questo è come filtrare l'immagine con un insieme di maschere. Le immagini risultanti dal filtraggio sono prese come feature della texture.

$$M_{00} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad M_{00} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad M_{00} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$M_{00} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad M_{00} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \quad M_{00} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

Tabella 2.2: Maschere corrispondenti ai momenti lungo una regione 3×3.

Le maschere si ottengono definendo una finestra di dimensione $W \times W$ e prendendo (i, j) come il punto dell'immagine in cui si calcolano i momenti, allora per ogni punto (m, n) ricadente nella finestra con centro (i, j), le normali si ottengono da:

$$x_m = \frac{(m-i)}{W/2} \; y_n = \frac{(n-j)}{(W/2)}$$

Allora i momenti in una finestra di centro (i, j) sono calcolati come:

$$m_{pq} = \sum_{n=-\frac{W}{2}}^{\frac{W}{2}} \sum_{m=-\frac{W}{2}}^{\frac{W}{2}} I(m, n) x_m^p y_n^q$$

I coefficienti della maschera sono dati dai coefficienti della sommatoria, per ogni pixel della finestra. Se R è una regione 3×3, allora le maschere che ne risultano sono quelle di Tabella 2.2.

Un altro approccio all'analisi in frequenza delle immagini di texture si riversa nel dominio di Fourier. Per passare all'analisi di Fourier, un'immagine viene scomposta e rappresentata nelle sue componenti frequenza e orientamento. La trasformata di Fourier è una analisi della frequenza globale contenuta in un segnale. Molte applicazioni richiedo che l'analisi sia localizzata in un dato dominio spaziale. Questo viene gestito introducendo dipendenze spaziali nell'analisi di Fourier, attraverso la *short-time Fourier transform*, ossia localizzando l'analisi all'interno di una finestra.

Si definisce la trasformata di Fourier nel breve periodo per un segnale unidimensionale come:

$$F_w(u, \xi) = \int_{-\infty}^{\infty} f(x) w(x - \xi) e^{j2\pi u x} dx$$

Qualora $w(x)$ corrisponde ad una Gaussiana la trasformata diventa la trasformata di Gabor.

Sono molti i lavori di ricerca svolti nel corso degli anni, per avere comunque un'idea di come possono essere diversi gli approcci per il riconoscimento delle texture può essere mensionato il lavoro di X. Liu e DL. Wang presentato in [26]. X. Liu e DL. Wang prendono come feature delle texture gli spettri degli istogrammi costruiti sulle risposte di una serie di filtri applicati all'immagine. Due pattern corrispondono alla stessa texture quando la distanza tra gli istogrammi è minima. Nell'articolo [26] viene spiegato quale è stata la procedura di selezione delle feature e tra i filtri usati risultano anche i filtri di Gabor. Sono di fatti i filtri più diffusi nell'analisi di texture perché producono buoni risultati, riuscendo a catturare le primitive che compongono la texture.

Un altro articolo interessante è [27] nel quale la classificazione delle texture avviene tramite il K-means clustering prendendo come centroide la risposta media dei filtri applicati all'immagine. La texture viene di fatto modellata tramite la probabilità congiunta delle risposte di una serie di filtri, dove la distribuzione probabilistica è rappresentata dalla frequenza degli istogrammi dei centroidi.

Questi articoli sono solo esempi di come le tecniche di riconoscimento dei pattern, già discusse nei precedenti paragrafi, possono essere integrate tra loro per il particolare caso delle texture, e forniscono dei validi suggerimenti sull'uso dei filtri nell'elaborazione di immagini ed estrazione di feature.

2.6 Osservazioni

Un problema di Pattern Recognition può essere affrontato con diversi approcci e usando tipi di classificatori diversi, a seconda dei dati che devono essere analizzati. Anche la rappresentazione dei dati cambia al variare dell'approccio seguito: si può dover operare sui pixel stessi dell'immagine guardando l'intensità rilevata, oppure su una rappresentazione diversa dell'immagine che mette in evidenza caratteristiche degli elementi appartenenti alla scena. Per effettuare il processo di riconoscimento il classificatore deve essere allenato usando un insieme di pattern campioni, detto training set, con l'analisi dei quali impara a distinguere i pattern che deve riconoscere e sotto quale

categoria classificarli. Questo processo di apprendimento può essere supervisionato, quindi ai campioni vengono associate delle etichette, oppure non supervisionato, in tal caso il classificatore deve apprendere dall'analisi stessa quanti e quali pattern deve distinguere.

Tenendo conto di tutte queste differenze si può comunque andare a generalizzare un classico processo di Pattern Recognition tramite lo schema in Fig. 2.2.

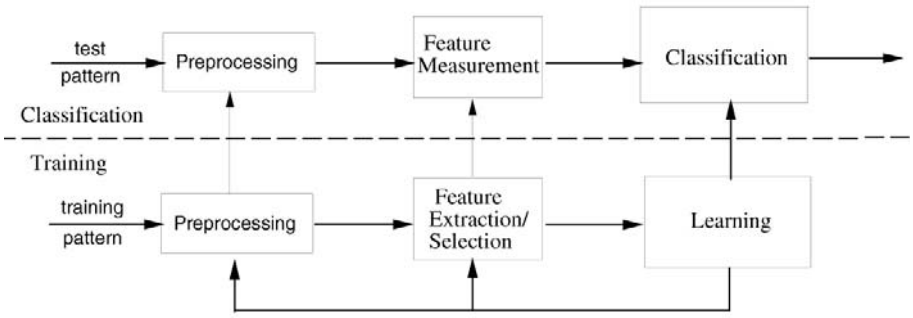


Figura 2.2: Modello di riconoscimento pattern.

Su questo modello si possono distingure due fasi del processo di riconoscimento: 1. il *training*, che riguarda tutte le attività rivolte all'allenamento del classificatore e 2. la *classificazione*, che rappresenta la fase di test in cui si va a verificare l'efficacia dell'apprendimento.

Nel modulo di "Preprocessing" l'immagine viene elaborata in modo da separare il pattern di interesse dallo sfondo, rimuovere eventuale rumore ed effettuare eventuali ulteriori operazioni, come la riduzione di scala, utili per velocizzare e migliorare i risultati delle fasi successive.

In modalità "Training" la selezione ed estrazione di feature cerca le caratteristiche più appropriate per la rappresentazione dei pattern di input e il classificatore viene allenato per partizionare lo spazio delle feature. Il risultato ottenuto permette al progettista di ottimizzare la fase di "Preprocessing" e le strategie usate nel "Feature Extraction/Selection".

In modalità "Classification" il classificatore precedentemente allenato assegna i pattern in ingresso da processare a una delle classi di pattern che ha catalogato nella fase di training sulla base delle misure delle feature estratte nel modulo "Feature Measurement".

Le performance di un classificatore dipendono sia dal numero di campioni disponibili per l'addestramento che dai valori stessi dei campioni e dall'insieme di feature che vengono prese in considerazione. L'obiettivo della progettazione di un sistema di riconoscimento è quello di riuscire a classificare nuovi campioni che saranno tipicamente diversi da quelli usati per l'addestramento. Perciò la proprietà che deve essere rispettata è la *generalità*: la capacità del classificatore di riconoscere le classi dei pattern di interesse tramite l'applicazione delle regole decisionali stabilite sulla base dei valori delle feature estratte nell'allenamento.

La poca generalità del classificatore può essere indotta da uno dei seguenti fattori: 1. il numero delle feature è troppo largo rispetto al numero dei campioni del training set - problema di dimensionalità - 2. il numero di parametri non noti associati al classificatore (caso di classificatori parametrici come classificatori polinomiali o reti neurali grandi) è troppo largo 3. un classificatore è troppo ottimizzato sul training set - overtrained o overfitting.

La probabilità di errore non aumenta con il numero delle feature, né è legata al numero dei campioni dell'addestramento, ma in pratica è stato osservato che l'aumento delle feature può degradare le performance se il numero dei campioni risulta essere troppo piccolo rispetto al numero delle feature. La dimensionalità delle feature incide sia sull'accuratezza del classificatore che sul costo delle misurazioni. Con un basso numero di feature, ma fortemente significativo, il classificatore risulterà più veloce, richiederà minor impiego di memoria e sarà abbastanza accurato. D'altro canto, una riduzione eccessiva del numero di feature può far perdere il potere discriminante e quindi diminuire l'accuratezza dei risultati. È per questo che viene posta molta attenzione alla fase di estrazione e selezione delle feature.

Il problema di *overfitting*, è il problema che si verifica quando l'errore prodotto in fase di training è troppo piccolo, in tal caso non si può pensare che sia un risultato positivo in quanto vorrebbe dire che il classificatore ha perso la sua generalità, si è adattato troppo ai dati di addestramento e in presenza di nuovi dati non sarà in grado di effettuare una buona classificazione.

Viste queste problematiche, nella progettazione di un classificatore occorre trovare un equilibrio tra il numero di campioni, il numero di feature e la complessità del classificatore.

È doveroso anche notare che la combinazione di più approcci e l'uso di più classificatori in parallelo o in cascata possa aiutare a migliorare i risultati.

Inoltre affrontare un problema scomponendolo in sottoproblemi e risolvendo questi con strategie diverse o, quando possibile, scomporre un pattern in sottostrutture, sono tutte strategie che possono aiutare a migliorare sia i costi computazionali che i risultati della classificazione.

2.6.1 Esempio

Un lavoro interessante di combinazione tra segmentazione, rilevamento e classificazione di pattern è stato svolto da Z. Tu, X. Chen, A.L. Yuille e SC. Zhu e descritto dettagliatamente nel [28].

Questo lavoro mette insieme molte delle tecniche e approcci descritti in questo capitolo, ed è un esempio interessante di collaborazione tra tutte queste metodologie con l'obiettivo di costruire un sistema efficace per la segmentazione delle immagini e il riconoscimento di texture e oggetti all'interno delle immagini.

Nel [28] viene descritto quale è il loro approccio e come combinano la segmentazione con il rilevamento e il riconoscimento di oggetti. Partono dall'idea di interpretare un'immagine con un grafico strutturato a livelli. Il nodo radice rappresenta l'intera scena, i nodi intermedi le regioni componenti un unico oggetto, mentre le foglie sono i singoli elementi che si possono individuare nell'immagine e che vengono aggregate per costruire un singolo oggetto. I nodi dello stesso livello sono legati da relazioni con le quali è possibile aggregare o disaggregare i nodi stessi. Il grafico può essere costruito in modalità top-down o bottom-up con mosse definite tramite le Catene di Markov. Le mosse non sono altro che operazioni di transizione che consentono la creazione o l'eliminazione di nodi, la divisione o l'unione di regioni, la modifica degli attributi di un nodo, la modifica dei confini di una regione. L'obiettivo di questo lavoro è quello di costruire un algoritmo per fare il parsing delle immagini tramite attività di composizione e decomposizione sull'immagine, per il riconoscimento di pattern. Il problema è stato formulato secondo l'approccio statistico, definendo per ogni possibile pattern un modello probabilistico sulla base di condizioni diverse, vista la diversa natura dei pattern. Infatti sono stati presi in considerazione regioni generiche come texture e zone d'ombra, e oggetti come volti e testi. Inoltre l'approccio per il modello di ciascun pattern cambia a seconda della procedura di visita (se bottom-up o top-down). Per esempio, in caso di visita bottom-up, per il riconoscimento dei caratteri, quindi per la formazione di testi, sono stati usati insiemi di

template rappresentanti numeri e lettere e formulate le probabilità tenendo in considerazione le possibili scalature, inclinazioni, deformazioni e forme; mentre per la generazione del modello dei volti è stato fatto uso del Principal Component Analysis (PCA) per ottenere la rappresentazione dei volti in termini di feature. Nel caso di visita top-down invece il riconoscimento dei testi e dei volti sfrutta le tecniche di boosting come AdaBoost[29][30], che sono particolari tecniche di apprendimento supervisionato che combinano in cascata i risultati di più classificatori, singolarmente non molto discriminanti, per costruire un classificatore più discriminante. Classificatori tipo AdaBoost e Haar-like[31] sono molto usati nel riconoscimento di volti e testi.

La strategia utilizzata dall'algoritmo di Tu, Chen, Yuille e Zhu può essere riassunta nei seguenti passi:

- il grafico viene costruito a partire dai singoli pixel, usati come foglie, successivamente vengono unite quelle regioni che sono ritenute simili in modo da costruire regioni atomiche dell'immagine;
- l'algoritmo procede analizzando queste regioni atomiche e raggruppandole per costituire regioni con la stessa texture, assegnandole ad un nodo del grafo;
- vengono calcolate le probabilità per stabilire se nodi tra loro vicini possono essere raggruppati costituendo un unico pattern. A questo punto vengono applicate le mosse di fusione o divisione tra nodi con l'algoritmo di Markov Chain Monte Carlo;
- per confermare il riconoscimento di particolari oggetti, come testi e volti, questa visita, di tipo bottom-up, viene affiancata dalla visita top-down che si ha tramite l'applicazione delle tecniche di riconoscimento AdaBoost. In questo modo si verifica una costruzione per espansione dei nodi del sottografo e non più per aggregazione; si parte dalla radice (volto) fino alle foglie (le componenti del volto).

L'approccio usato da Z. Tu, X. Chen, A.L. Yuille e SC Zhu, mostra buoni risultati anche se poco performante in termini computazionali. È un interessante esempio di fusione tra vari approcci e tecniche per la risoluzione di problemi di Pattern Recognition e mostra come l'integrazione di operazioni usualmente usate per la risoluzione di problemi diversi tra loro, quali in

questo caso la segmentazione di immagini e il riconoscimento di oggetti, può invece portare a buoni risultati in termini di efficacia.

2.7 Conclusioni

In questo capitolo sono state presentate le discipline toccate durante lo svolgimento del presente lavoro di tesi, quali la Computer Vision, la Pattern Recognition e il Machine Learning.

Sono state presentate le principali tecniche di risoluzione dei problemi di Pattern Recognition, gli approcci noti in letteratura a questi tipi di problemi e i classificatori più noti e diffusi, ponendo un particolare interesse sulle tecniche utilizzate nel riconoscimento della *Posidonia oceanica* e che verranno riprese nel capitolo successivo.

È stata inoltre presentata la terminologia propria di queste discipline che sarà utile per la comprensione del lavoro svolto.

È stato affrontato il processo di riconoscimento dei pattern partendo dall'acquisizione dei dati e inquadrando le problematiche che vengono incontrate nelle scelte della loro rappresentazione fino ad arrivare alla presentazione di varie tipologie di classificatori.

I classificatori enunciati in questo capitolo non sono comunque gli unici, e non sempre vengono usati in modo disgiunto; è stato infatti fatto notare durante l'esposizione che la realizzazione di un classificatore può in realtà prevedere l'uso di più classificatori e uno stesso problema può essere affrontato con la collaborazione di approcci diversi.

CAPITOLO 3

Riconoscimento della *Posidonia oceanica*

Questa tesi è stata motivata dalla necessità di capire se era possibile sviluppare un sistema che in modo automatico riusciva a riconoscere la *Posidonia oceanica* all'interno di fotografie di fondali marini. Si pongono quindi i seguenti interrogativi:

1. È possibile formulare un modello rappresentativo del pattern Posidonia?
2. Quale approccio usare per la risoluzione dello specifico problema di Pattern Recognition?
3. Quale classificatore usare?

È stato svolto un lavoro di ricerca e sperimentazione per trovare la risposta a queste domande.

In questo capitolo verranno presentate le scelte prese per la risoluzione del problema, l'approccio utilizzato, il modello costruito per il pattern Posidonia e il classificatore usato per il suo riconoscimento. Infine verranno forniti i dettagli implementativi del sistema di riconoscimento proposto.

3.1 Tecnica di Riconoscimento

Per definire il problema occorre collocarsi nello scenario in cui ci sia una imbarcazione sulla quale è montata una macchina fotografica appena sotto

la superficie del mare e perpendicolarmente ad essa, e che scatta fotografie georeferenziate del fondale marino. Le immagini raccolte devono essere analizzate una per una per determinare in ciascuna la presenza o meno della *Posidonia oceanica* e collocare nello spazio tale informazione, sulla base delle coordinate associate alla foto; per avere l'opportunità di mappare l'estensione del posidonieto e proseguire con le dovute analisi sul limite inferiore, superiore ed eventuali altre misurazioni, al fine di monitorare lo stato della *Posidonia oceanica*.

L'unica informazione che una semplice fotografia ci comunica è il colore. Si pone quindi il problema di distinguere la Posidonia da tutto il resto, sulla base di questa unica informazione. Non si può affermare che le praterie di Posidonia costituiscano un insieme geometricamente ben definibile, né si può procedere con un approccio statistico visto lo scarso numero delle immagini raccolte in appendice A e la loro eterogeneità. La definizione del colore, della forma e delle strutture interne della prateria variano con la profondità del fondale, con la sua inclinazione, con la limpidità dell'acqua, con la luminosità esterna e con la direzione delle correnti che influenzano l'orientamento delle foglie. Occorre perciò cercare una qualche rappresentazione del modello della Posidonia che risulti invariante nei confronti di tutte queste variabili.

Ricordandoci che le praterie sono costituite da foglie nastriformi, prendendo una finestra sufficientemente piccola sulla prateria ad una certa scala e osservando gli insiemi dei campioni raccolti in appendice A, allora si può pensare di interpretare la prateria di *Posidonia oceanica* come costituita da una "trama". Da questa trama, sarà necessario estrarre le caratteristiche strutturali che costituiranno la rappresentazione della classe Posidonia.

L'algoritmo che viene adoperato suddivide l'immagine da analizzare in finestre grandi come l'immagine della trama prescelta, facendole scorrere ogni volta di un pixel. Per ogni finestra viene eseguita una operazione di confronto con il prototipo, dove per prototipo si intende la rappresentazione della classe Posidonia in termini di feature. Il risultato della operazione di confronto per ogni finestra esprime la presenza o l'assenza della Posidonia in quel frammento di immagine.

Si pone quindi il problema di determinare a quale livello di scala estrarre l'immagine della trama e quale deve essere la dimensione della finestra, in modo da 1. non appesantire il carico computazionale, dimensioni troppo piccole implicano maggiori operazioni di confronto; 2. affinare l'accuratezza del

risultato, dimensioni troppo grandi forniscono una approssimazione eccessiva.

Figura 3.1: Immagine template.

Sono state testate una serie di immagini a partire da dimensioni 150 × 150 pixel fino a giungere al prototipo di Fig. 3.1 di 30 × 30 pixel con la quale, nonostante la sua limitata grandezza, sono stati riscontrati risultati sufficientemente accurati senza eccessivi costi computazionali.

3.1.1 Modello del pattern Posidonia

Il modello formulato per la rappresentazione del pattern Posidonia è stato costruito con un approccio di tipo misto: sintattico, statistico. Si è cercato dapprima, una tecnica per l'estrazione delle caratteristiche strutturali, sull'immagine template, tramite l'applicazione di filtri, e successivamente di estrarre la componente statistica del pattern, tramite il confronto tra gli spettri degli istogrammi costruiti sulle immagini ottenute dalle convoluzioni.

L'appartenenza alla classe Posidonia è stabilita dalla predizione del classificatore, che analizza, per ogni filtro, la distanza degli spettri degli istogrammi calcolati sull'immagine template di Fig. 3.1 e l'immagine in fase di analisi, successivamente all'operazione di convoluzione.

Filtri

Per l'estrazione delle feature sono stati testati, su un insieme di campioni, una serie di filtri e matrici di convoluzione. Sono stati selezionati quelli che hanno fornito i migliori risultati.

Facendosi guidare dalla letteratura [26], [32], sono stati testati i filtri di Gabor, tramite l'applicazione della funzione (3.1), con una serie di valori per l'orientamento e la frequenza. I kernel di Gabor, come già accennato nella Sezione 2.5, sono frequentemente usati per la loro capacità di lavorare efficientemente su diversi tipi di pattern naturali e di garantire l'invarianza nei confronti di orientamento del pattern e scalature.

$$G_{\lambda,\theta,\varphi,\sigma,\gamma}(x,y) = e^{-\frac{\bar{x}^2+\gamma^2\bar{y}^2}{2\sigma^2}} \cos(2\pi\frac{\bar{x}}{\lambda}+\varphi) \tag{3.1}$$

dove

$$\bar{x} = x\cos\theta + y\sin\theta$$

$$\bar{y} = -x\sin\theta + y\cos\theta$$

La funzione di Gabor (3.1) impiegata nel sistema di riconoscimento ha prodotto buoni risultati con un valore di σ pari ad 1, sono risultati utili kernel con centri simmetrici quindi $\varphi = 0$, invece il valore ottimo per γ è risultato essere 0.5, per indicare centri di tipo allungato. I valori di x e y sono stati presi nel range $[-3, 3]$, creando kernel di Gabor 7×7. Gli unici parametri rimasti risultano essere θ, che indica l'orientamento del centro, e λ che indica la lunghezza dell'onda della componente coseno. Sono stati testati varie combinazioni di θ e λ fino ad ottenere $\Psi = \{g_1, g_2, g_3, g_4, g_5\}$, insieme dei filtri usati per costruire il modello, composto da:

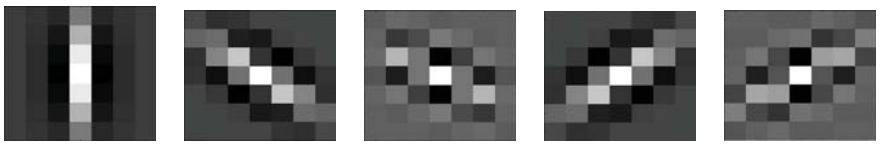
g_1**:** kernel di Gabor con $\theta = 0$ e $\lambda = 3$ in Fig. 3.2a;

g_2**:** kernel di Gabor con $\theta = -180$ e $\lambda = 3$ in Fig. 3.2b;

g_3**:** kernel di Gabor con $\theta = 90$ e $\lambda = 2$ in Fig. 3.2c;

g_4**:** kernel di Gabor con $\theta = 180$ e $\lambda = 3$ in Fig. 3.2d;

g_5**:** kernel di Gabor con $\theta = 45$ e $\lambda = 2$ in Fig. 3.2e.



(a) (b) (c) (d) (e)

Figura 3.2: Insieme Ψ dei kernel di Gabor usati.

L'immagine di Fig. 3.1 viene filtrata con ciascuno dei filtri $g_i \in \Psi$ e sul risultato del filtraggio viene calcolato l'istogramma. Il prototipo per il pattern Posidonia è ora costituito dal vettore degli istogrammi delle immagini risultate dal filtraggio, $\mathbf{H}_T = (H_T^1, H_T^2, H_T^3, H_T^4, H_T^5)$.

Vettore delle feature

Presa una finestra W sull'immagine campione ed applicati su di essa gli $n = 5$ filtri appartenenti a Ψ, si ricava dalle immagini filtrate il vettore degli isto-

grammi per quella finestra, $\mathbf{H}_W = (H_W^1, H_W^2, \ldots, H_W^n)$; essendo, gli spettri degli istogrammi, delle distribuzioni probabilistiche è possibile definire una distanza tra due istogrammi, perciò si calcola la distanza tra il modello di riferimento, $\mathbf{H}_T$, e le distribuzioni della finestra, $\mathbf{H}_W$, per ogni immagine filtrata, tramite la stima della distribuzione χ^2 secondo cui la distanza tra due istogrammi è data da

$$\chi^2(H_T^i, H_W^i) = \sum_h \frac{(H_T^i(h) - H_W^i(h))^2}{H_T^i(h) + H_W^i(h)}.$$

Da qui si ottiene il vettore delle feature per la finestra presa in esame:

$$\mathcal{F}_{\mathcal{W}} = \begin{bmatrix} \chi^2(H_T^1, H_W^1) \\ \vdots \\ \chi^2(H_T^n, H_W^n) \end{bmatrix}$$

Nelle Tabelle 3.1 e 3.2 sono proposti alcuni esempi di valori assunti dalle feature per campioni appartenenti alla classe Posidonia (Tabella 3.1) e per campioni non appartenenti alla classe Posidonia (Tabella 3.2).

$$f_1 = \begin{bmatrix} 0.46 \\ 0.44 \\ 0.43 \\ 0.42 \\ 0.43 \end{bmatrix} \quad f_3 = \begin{bmatrix} 0.66 \\ 0.60 \\ 0.94 \\ 0.68 \\ 0.75 \end{bmatrix} \quad f_5 = \begin{bmatrix} 0.84 \\ 1.15 \\ 0.94 \\ 0.91 \\ 0.97 \end{bmatrix} \quad f_7 = \begin{bmatrix} 0.90 \\ 1.20 \\ 1.59 \\ 0.89 \\ 1.57 \end{bmatrix}$$

Tabella 3.1: Esempi di vettori delle feature per campioni positivi.

$$f_2 = \begin{bmatrix} 1.99 \\ 1.99 \\ 2 \\ 2 \\ 2 \end{bmatrix} \quad f_4 = \begin{bmatrix} 1.77 \\ 1.75 \\ 2 \\ 1.88 \\ 2 \end{bmatrix} \quad f_6 = \begin{bmatrix} 1.56 \\ 1.57 \\ 1.97 \\ 1.65 \\ 1.90 \end{bmatrix} \quad f_8 = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 2 \\ 2 \end{bmatrix}$$

Tabella 3.2: Esempi di vettori delle feature per campioni negativi.

Per sottolineare la differenza che sussiste tra i campioni di Tabella 3.1 e quelli in Tabella 3.2 possiamo andare a calcolare la media tra f_1, f_3, f_5, f_7 e confrontare il vettore medio che otteniamo, $v = [0.72, 0.85, 0.97, 0.73, 0.93]$, con ciascuno dei campioni in Tabella 3.1 e in Tabella 3.2 tramite la norma Euclidea della differenza vettoriale, $\|f_i - v\| \, \forall i = 1, \ldots, 8$. Il risultato è mo-

strato nel grafico di Fig. 3.3, dove possiamo notare la distanza che intercorre tra campioni positivi e campioni negativi.

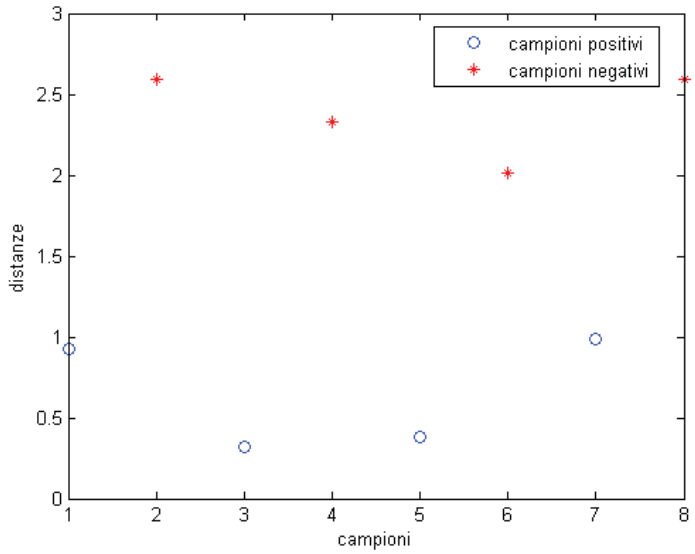


Figura 3.3: Proiezione della norma Euclidea dei vettori delle feature.

3.1.2 Il Classificatore

Il problema di Pattern Recognition che abbiamo è essenzialmente quello di separazione tra due insiemi. Definiamo positivi i pattern appartenenti alla classe Posidonia e negativi i pattern che non appartengono alla classe Posidonia. È necessario definire e insegnare al classificatore tramite un addestramento supervisionato, quali sono i pattern positivi e quale sia la regola di separazione tra positivo e negativo.

Pensando alle caratteristiche di questo problema, che ricerca la separazione tra due classi di oggetti, osservando i valori delle feature assunti dai campioni positivi e negativi (alcuni esempi sono in Tabella 3.1 e Tabella 3.2), e riportando alla mente l'esposizione dei classificatori di Sezione 2.4.1, sembrerebbe adatta al caso Posidonia la classificazione con le Support Vector Machine.

Difatti, facendo riferimento al grafico di Fig. 3.3, i dati sembrerebbero facilmente separabili vista la grande varianza che intercorre tra campioni positivi e campioni negativi. Si può tentare quindi di ricercare, se esiste,

un iperpiano di separazione che minimizzi il rischio di classificazione tramite la massimizzazione del margine tra valori positivi e valori negativi, come espresso nella funzione (2.8).

È stato perciò impiegato un classificatore di tipo SVM, con l'uso di una funzione kernel lineare per la risoluzione della (2.9), che ha prodotto buoni risultati di classificazione.

Il classificatore è stato allenato con un insieme di 48 immagini di 150×150 pixel, raccolte in appendice A.1, dalle quali sono stati estratti i vettori delle feature passati al classificatore come insieme buono per la rappresentazione del pattern Posidonia.

Si può verificare l'abilità del classificatore ottenuto dai risultati sulle immagini riportate in Tabella 3.3.

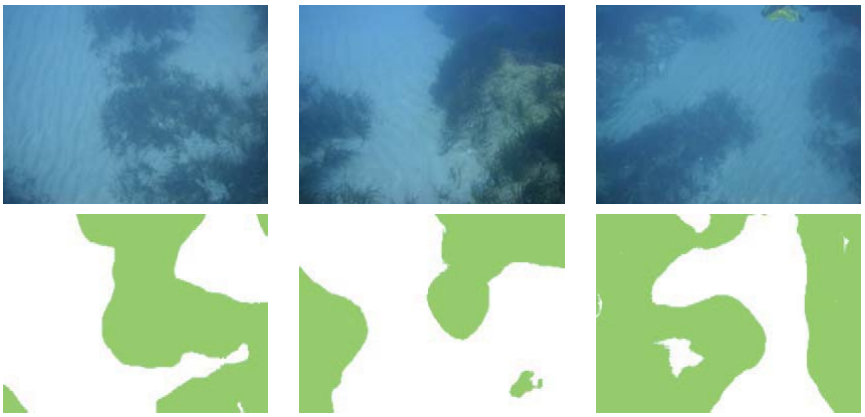


Tabella 3.3: Esempi di riconoscimento su immagini reali.

3.2 Implementazione

Costruito il modello e determinato il tipo di classificatore da usare, il sistema di riconoscimento automatico della *Posidonia oceanica* è stato implementato ed integrato in una possibile applicazione con la quale vengono svolte le semplici attività di riconoscimento su immagine e di visualizzazione.

3.2.1 Strumenti

Il codice è stato sviluppato in C^{++} e per l'elaborazione delle immagini è stata scelta la libreria OpenCV[1] versione 2.0 vista la sua notorietà nel mondo dell'Analisi e dell'Elaborazione delle immagini, la sua facile integrazione e l'ampia scelta di algoritmi e funzionalità che fornisce.

È stato usato NetBeans[2] 6.9 come ambiente di sviluppo. Il sistema operativo utilizzato è stato Windows Xp e questo ha reso necessario l'ausilio, per la compilazione del codice, dell'ambiente Cygwin[3] (versione 1.7.5) come richiesto da specifiche OpenCV per la versione 2.0.

Inoltre è stata fornita un'interfaccia grafica come ausilio per i test delle funzionalità del sistema di riconoscimento implementato. Anche per questa il codice è stato sviluppato in C^{++} usando come libreria grafica Qt[4] versione 4.6, in ambiente di sviluppo QtCreator.

OpenCV

OpenCV è l'acronimo per Open Source Computer Vision, è una libreria che raccoglie una vasta scelta di algoritmi per sistemi di visione real time. La versione corrente è la 2.1 ed è distribuita sotto licenza BSD [5], sia per piattaforma Linux che Windows con Visual Studio 2008 o con ambiente Linux-like. È la libreria più diffusa nella comunità dell'Analisi ed Elaborazione delle immagini. Fornisce più di 500 algoritmi divisi su più moduli; sono implementati i maggiori operatori morfologici, tra cui "chiusura" e "apertura", di convoluzione e trasformazione, tra cui gli operatori di Sobel, Laplace, Canny, le trasformate di Hough, funzioni per la trasformazione dell'immagine tra coordinate cartesiane e polari, calcolo e operazioni di confronto degli istogrammi, l'algoritmo di Template Matching e molte altre; include inoltre una libreria ben fornita di algoritmi per il Machine Learning tra cui il boosting, gli alberi decisionali, l'algoritmo di Expectation-maximization, K-nearest neighbor, classificatori Bayesiani, Reti Neurali e Support Vector Machine. Elabora immagini statiche e anche sequenze di frame provenienti da telecamere. È molto

[1] http://opencv.willowgarage.com/wiki/Welcome fa già riferimento alla versione corrente.

[2] http://netbeans.org/

[3] http://cygwin.com/

[4] sito ufficiale http://qt.nokia.com/ documentazione per la libreria reperibile su http://doc.qt.nokia.com/4.6/

[5] Open Source Initiative OSI http://opensource.org/licenses/bsd-license.php

diffusa non solo per la grande quantità di funzioni che fornisce ma anche per il basso costo computazionale e per la larga scelta di linguaggi con i quali è possibile integrare la libreria, infatti oltre che il C e C^{++}, sono stati forniti wrapper per Python e Java. Ovviamente bisogna tenere di conto che non tutte le funzionalità aggiunte nelle più recenti versioni sono già disponibili anche per i linguaggi diversi dal C/C^{++}. Per una guida all'uso della libreria oltre alla documentazione presente in rete reperibile dal sito ufficiale (http://opencv.willowgarage.com/wiki/Welcome) si può fare riferimento a [33].

3.2.2 Struttura del Codice

Il codice prodotto risulta diviso su due moduli, un modulo per il processo di riconoscimento e un modulo per la Graphic User Interface (GUI) dell'applicazione.

Il modulo del processo di riconoscimento è suddiviso su 4 file di header, per la definizione delle classi e le dichiarazioni delle funzioni definite negli altri 4 file `.cpp`, infine c'è il `main` dove sono riportate le funzioni usate per i test e il corpo del programma. La documentazione relativa alle classi implementate nel modulo di riconoscimento è stata riportata in Appendice B.

Il modello della Posidonia che è stato descritto precedentemente nella Sezione 3.1, è stato implementato nella classe `PosidoniaModel` la quale mantiene un costante riferimento all'oggetto immagine di tipo `IplImage` che rappresentante il template di Fig. 3.1. Inoltre tra le sue proprietà ritroviamo il riferimento all'array di istogrammi, di tipo `CvHistogram`, che rappresentano il prototipo per il pattern Posidonia come descritto in Sezione 3.1. Dopo aver creato un oggetto di tipo `PosidoniaModel` è necessario invocare il metodo `initModel` sull'oggetto per poter avviare le attività di inizializzazione del modello, ossia la creazione delle suddette proprietà, "immagine template" e "istogrammi". Il classificatore può essere invece allenato in seguito ad una richiesta dell'utente.

Ci siamo infatti collocati nello scenario in cui l'utente possa voler personalizzare l'allenamento del classificatore con un insieme di immagini che lui stesso ha l'opportunità di campionare.

Questa scelta è stata guidata dalla necessità di far corrispondere la tipologia dei campioni dell'allenamento alle immagini che si andranno ad analizzare in seguito, per quanto riguarda luminosità e profondità del fondale. Infat-

ti personalizzando l'allenamento del classificatore sulle condizioni ambientali ritrovate nel momento della campionatura delle immagini da analizzare, si ottiene di fatto un migliore risultato di classificazione. È comunque possibile far allenare il classificatore con un insieme predefinito di campioni, qualora non ci fossero a disposizione campioni ad-hoc.

Il classificatore è definito esso stesso come membro della classe `PosidoniaModel` ed è un oggetto di tipo `CvSVM`.

Fanno parte della classe `PosidoniaModel` i metodi per eseguire le operazioni di filtraggio, di allenamento del classificatore, di costruzione delle feature e di calcolo della distanza tra istogrammi. In un file separato, chamato `utilities`, sono raggruppate tutte le funzioni di utilità non strettamente legate al modello del pattern Posidonia, come per esempio la scansione di una directory per il ritrovamento di immagini, oppure la funzione per eseguire il ridimensionamento delle immagini o per creare l'istogramma a partire da un'oggetto immagine.

L'altra classe che è stata collocata all'interno del modulo di riconoscimento è `TrackMap`. Questa classe consente di tenere traccia delle informazioni relative ad una zona geografica sulla quale si stanno compiendo le attività di analisi.

Man mano che viene analizzata una immagine per il riconoscimento della *Posidonia oceanica*, il risultato della classificazione viene passato all'oggetto di tipo `TrackMap` tramite l'invocazione del metodo `add(IplImage * img, float nw_x, float nw_y)`, il quale richiede come parametri, oltre all'immagine, anche le coordinate di corrispondenza. `TrackMap` sarebbe quindi la classe per rappresentare una mappatura delle praterie di *Posidonia oceanica*. Consente di tenere traccia delle informazioni di presenza/assenza di Posidonia in una data area tramite la memorizzazione di queste informazioni all'interno di una struttura dati chiamata `subMap`.

`subMap` è definita come `typedef map<string,IplImage *> subMap`, ed accetta come chiave valori di tipo `string` e come oggetto riferimenti a `IplImage`. La chiave sarà la stringa della composizione tra latitudine e longitudine, separate da una virgola, alla quale corrisponde l'immagine che viene inserita. L'immagine che si inserisce in `subMap` corrisponde ad una cella della mappa. La mappa è suddivisa in una griglia immaginaria ed ogni cella della griglia è la minima unità che può essere inserita od estratta.

Costruzione del modello e riconoscimento

Il modello del pattern Posidonia viene inizializzato tramite l'invocazione di `initModel()` che provvede a costruire l'array contenente gli istogrammi delle immagini risultato delle operazioni di filtraggio con l'impiego di Ψ sulla Fig. 3.1. I filtri impiegati sono esattamente quelli descritti in Sezione 3.1.

La funzione di Gabor (3.1) è stata implementata con il seguente codice:

```
void PosidoniaModel::G(CvMat* a,double theta, double lambda){
    double gamma = 0.5;
    double psi = 0;
    double sigma_x = 1;
    double sigma_y = sigma_x/gamma;
    int val[7] = {-3,-2,-1,0,1,2,3};
    for(int i = 0; i < 7; i++){
        int x = val[i];
        for(int j = 0; j < 7; j++){
            int y = val[j];
            double x_theta=x*cos(theta)+y*sin(theta);
            double y_theta=-x*sin(theta)+y*cos(theta);
            double el = exp(-0.5*(pow(x_theta,2)/pow(sigma_x,2)
                                +pow(y_theta,2)/pow(sigma_y,2)))*
                            cos(2*PI/lambda*x_theta+psi);
            cvmSet(a,i,j,el);
        }
    }
}
```

Questa funzione prende come parametro la matrice nella quale va ad inserire gli elementi ricavati dall'applicazione della funzione (3.1), `theta` e `lambda` che indicano rispettivamente l'orientamento (θ) e la lunghezza dell'onda della componente coseno (λ).

Una volta creato il modello è possibile creare ed allenare il classificatore.

Come detto precedentemente il classificatore può essere allenato con un insieme predefinito di campioni, come quelli raggruppati in appendice A.1, oppure con un insieme di campioni a scelta dell'utente. L'allenamento del classificatore viene effettuato con l'invocazione di `train(vector<IplImage *> samples)`, che prende esattamente l'insieme dei campioni ritenuti buoni per l'allenamento. Una volta costruito il vettore delle feature e settati i parametri corretti, il classificatore può essere allenato:

```
...
CvTermCriteria criteria;
criteria=cvTermCriteria(CV_TERMCRIT_EPS,10000000,FLT_EPSILON);
CvSVMParams params=CvSVMParams(CvSVM::ONE_CLASS,CvSVM::LINEAR,
    1.0, 1.0, 1.0, 20.0, 0.5, 0.1, NULL, criteria);
...
svm->train(moresamples,moreack,0,0,params);
```

`moresamples` è una matrice che contiene tante righe quanti sono i campioni e tante colonne quanti sono i filtri. Ogni elemento (i, j) della matrice corrisponde alla distanza del campione i per l'immagine filtrata con il filtro j, dallo stesso filtro applicato all'immagine template. Tale distanza viene calcolata con l'invocazione di `cvCompareHist(this->hists_pos[filter],hist,CV_COMP_CHISQR )` dove `this->hists_pos` è l'array degli istogrammi sul template, `hist` è l'istogramma sul campione dopo il filtraggio e il flag `CV_COMP_CHISQR` indica proprio che ci interessa calcolare la distanza con l'uso della statistica χ^2.

Una volta allenato il classificatore, si può proseguire con le operazioni di analisi. L'analisi viene eseguita su una immagine alla volta tramite l'invocazione del metodo `predictPos(IplImage * src, IplImage * dst)` appartenente alla classe `PosidoniaModel`. `src` è l'immagine da analizzare, mentre `dst` è l'immagine di output che conterrà l'informazione presenza/assenza della Posidonia e la cui dimensione in pixel è ottenuta da:

larghezza larghezza di `src` - larghezza dell'immagine template + 1;

altezza altezza di `src` - altezza dell'immagine template + 1;

in quanto le finestre sull'immagine `src` sulle quali viene effettuata l'operazione di confronto, come descritto in Sezione 3.1.1, devono essere grandi come l'immagine prototipo e scorrono ogni volta di un pixel.

La predizione viene eseguita tramite l'invocazione, sull'oggetto classificatore, del metodo `svm->predict(feature)`, che a partire dal vettore delle feature restituisce 0 se il vettore appartiene al pattern Posidonia, 1 altrimenti. Perciò per la finestra (i, j) viene memorizzata in `dst` al pixel (i, j) l'informazione `DATA_POS`, di tipo `CvScalar` e valore `cvScalar(125,255,125)`, che indica la presenza di Posidonia se la predizione è 0, altrimenti inserirà `DATA_EMPTY`, che assume il valore `cvScalar(255,255,255)`, per indicare l'assenza.

L'immagine di output viene successivamente ridimensionata alle dimensioni originali per poi essere inserita nella mappa.

GUI

Per questo sistema di riconoscimento è stata proposta una interfaccia grafica minimale con il semplice scopo di testare il sistema, ma allo stesso tempo fornire un'idea di una possibile applicazione del sistema di riconoscimento.

Le attività che possono essere svolte sono:

- analizzare in modo automatico una serie di immagini raccolte;
- ottenere e visionare la mappa relativa all'area perlustrata;
- salvare tutte le informazioni ottenute sull'area perlustrata;
- riprendere le informazioni di un'area precedentemente esplorata;
- personalizzare l'allenamento del classificatore per migliorare la qualità dei risultati.

L'ultimo punto è particolarmente importante nei casi in cui l'utente si accorga che la zona è in condizioni ambientali difficoltose, per esempio ha una torbidità dell'acqua che compromette i risultati della classificazione. In tal caso c'è la possibilità di effettuare una raccolta di immagini e sottoporle a campionatura per la validazione da parte dell'utente.

In Fig. 3.4 viene mostrata l'interfaccia con la quale l'utente può selezionare i campioni buoni per l'allenamento.

L'attività di analisi viene fatta partire da una directory scelta dall'utente e, una volta avviata, occorre aspettare la terminazione prima di procedere con altre operazioni. Non si è immaginato uno scenario che necessiti risposte in tempo reale per questa interfaccia ma non si esclude che si possa adottare lo stesso sistema di riconoscimento in applicazioni real-time.

Appena avviata l'applicazione l'unica operazione che l'utente può svolgere, a parte aprire un file già esistente, è quella di creare una nuova mappa, indicando i riferimenti geografici (Fig. 3.5) della cartina, inserendo un nome per quella zona ed eventuali altre note. La mappa può essere poi interamente visualizzata a schermo nel pannello dell'applicazione grafica. Solo dopo la creazione della cartina è possibile procedere con tutte le altre operazioni. Questo per garantire l'inizializzazione all'avvio di tutte le istanze coinvolte nel modello.

All'avvio viene automaticamente effettuato l'allenamento del classificatore a partire dai campioni di appendice A.1. Ma in seguito sarà sempre

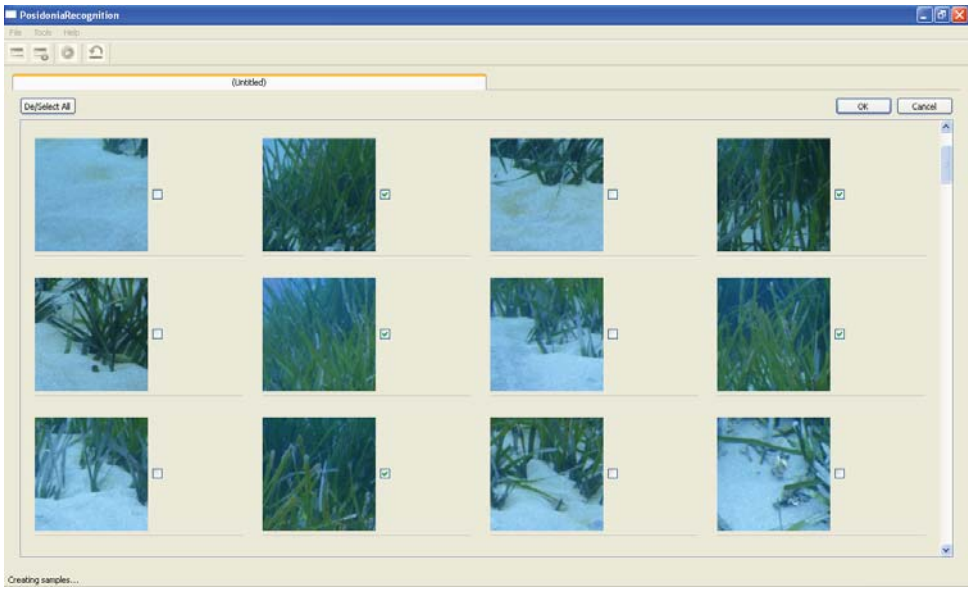


Figura 3.4: Pannello di selezione dei campioni buoni.

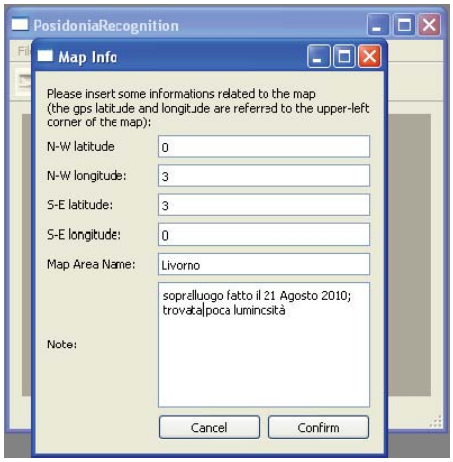


Figura 3.5: Pannello per l'insersione delle informazioni sulla mappa.

possibile per l'utente riallenare il classificatore con campioni diversi, quindi qualora spostandosi si noti una diversità delle condizioni dell'acqua si può scegliere di modellare il classificatore sulle proprie esigenze.

Dall'interfaccia grafica è inoltre possibile richiedere il salvataggio di tutte le informazioni raccolte, come anche riaprire mappe create in tempi precedenti.

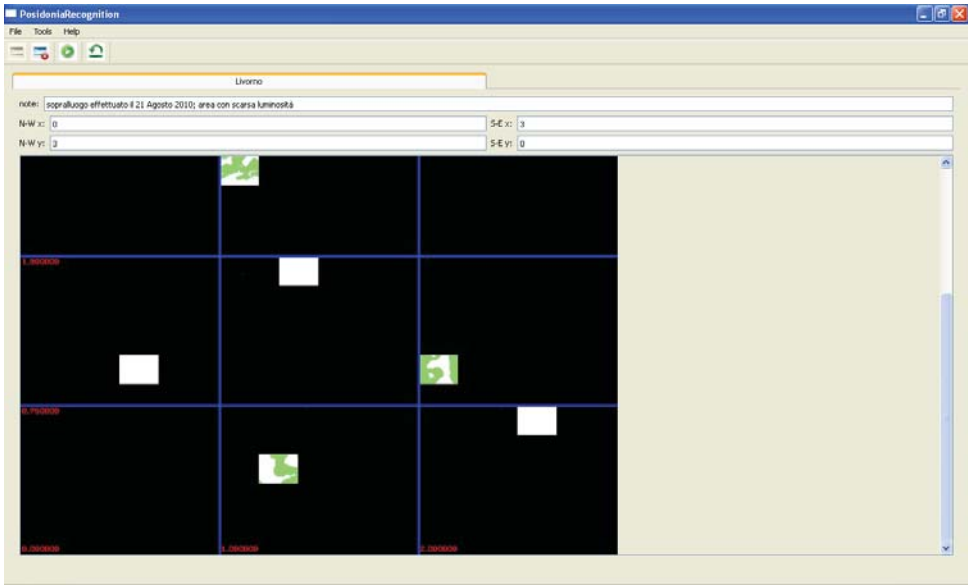


Figura 3.6: Interfaccia del sistema con la visualizzazione della mappa dopo l'analisi.

3.2.3 Particolari scelte implementative

Persistenza dei dati

I dati della mappatura della *Posidonia oceanica* vengono salvati in file di tipo `xml`. Questa scelta è stata guidata dalla facilità di costruzione e interpretazione dei file xml, dalla diffusione di questo stesso linguaggio per il salvataggio dei dati in varie applicazioni e anche dall'uso della libreria OpenCV che già contiene un modulo per garantire la persistenza dei dati traducendo i propri oggetti, con le loro proprietà, in nodi di un albero xml con radice `<opencv_storage>`. Ci si affida perciò completamente al modulo di persistenza dei dati della libreria.

Gli oggetti coinvolti nel salvataggio e recupero sono esclusivamente l'oggetto di tipo `TrackMap`, per mantenere traccia delle analisi fatte, e il classificatore `CvSVM` usato per l'ultima analisi fatta. Non occorre tenere traccia del modello del pattern Posidonia in quanto invariante del processo di riconoscimento e quindi ricreato ad ogni avvio del sistema.

Il classificatore viene salvato invocando direttamente la funzione `save` sull'oggetto `svm`. L'oggetto viene annidato nell'albero xml sotto il nodo `<model_classifier_1 type_id="opencv-ml-svm">`.

Per la classe `TrackMap` vengono creati i nodi `<trackmap_coord_nordwest>`, `<trackmap_coord_southest>`, `<trackmap_note>` e `<trackmap_name>`, nei quali vengono memorizzate le proprietà dell'oggetto.

Infine per tenere traccia dei risultati di analisi memorizzati in `subMap` viene incluso il nodo `<images_key>` dove vengono inserite le chiavi e a seguire nel nodo `<images_file>` compaiono, in corrispondenza con le coordinate contenute in `<images_key>`, i cammini dei file delle immagini delle celle.

Infatti le celle vengono salvate in formato JPG per rendere più leggero il file `.xml` e allo stesso tempo si consente di visionare le singole aree anche da semplici programmi di visualizzazione immagini.

Comunicazione tra processi

Il modulo di riconoscimento e quello dell'interfaccia verso l'utente sono stati sviluppati separatamente avendo due obiettivi completamente diversi; il primo è stato implementato per verificare e dimostrare la correttezza del modello costruito per il pattern Posidonia, mentre l'applicazione grafica ha puramente il compito di fornire un'idea applicativa del sistema di riconoscimento sviluppato.

Dai primi test effettuati era risultato inoltre, che l'integrazione dei due codici apportasse un forte rallentamento nell'analisi delle immagini da parte del modulo di riconoscimento. In principio, per integrare i due codici è stato necessario ricompilare la libreria OpenCV con compilatore MinGW[6], usato per la compilazione dell'interfaccia grafica, ma questa operazione aveva causato un forte rallentamento nell'esecuzione delle funzioni della libreria per l'elaborazione delle immagini.

[6] http://www.mingw.org/

Onde evitare questo problema e per mantenere la logica della separazione tra i due moduli, sono stati prodotti due programmi, uno per il modulo di riconoscimento e uno per l'interfaccia grafica, che danno vita a due processi disgiunti e comunicanti tramite PIPE.

Per le comunicazioni vengono utilizzate due PIPE, una per la richiesta di operazioni da parte dell'interfaccia verso il modulo di riconoscimento, in tal caso l'applicazione grafica rappresenta il lato client della comunicazione mentre il modulo di riconoscimento è il lato server. Qualora l'applicazione grafica necessiti, prima di continuare con la sua esecuzione, di una risposta allora questa si pone in attesa su una seconda PIPE.

Coesistono quindi sia comunicazioni di tipo sincrono che asincrono tra i due processi.

Le operazioni sincrone sono:

- richiesta dell'immagine mappa;
- richiesta della creazione di campioni;
- richiesta di apertura di un file .xml;
- richiesta di analisi delle immagini da directory.

Mentre le operazioni di salvataggio, creazione della mappa e allenamento del classificatore sono invece di tipo asincrono.

In Fig. 3.7 è schematizzato il ciclo di vita del processo per il modulo di riconoscimento. Dopo aver inizializzato il modello della Posidonia e creato ed aperto in lettura la PIPE, il processo rimane in attesa fino a che non riceve una qualche richiesta dal client. La stringa che legge da PIPE sarà costituita da un valore indicante l'operazione e a seguire una serie di parametri che variano a seconda del tipo di operazione richiesta. Gli elementi della stringa sono divisi dal token "\\".

Le richieste che può ricevere sono codificate con un valore intero che varia tra 0 e 8. Se `op` è uguale a 0 allora il processo deve terminare, ciò accade quando l'interfaccia grafica ha ricevuto la richiesta di chiudersi; in tal caso prima di terminare indica al processo di riconoscimento di terminare a sua volta. Nei casi in cui `op` sia diverso da 0, allora una delle seguenti operazioni deve essere eseguita:

1. deve essere eseguito l'allenamento automatico. L'utente non ha indicato nessun insieme di campioni perciò si procede con l'allenamento del

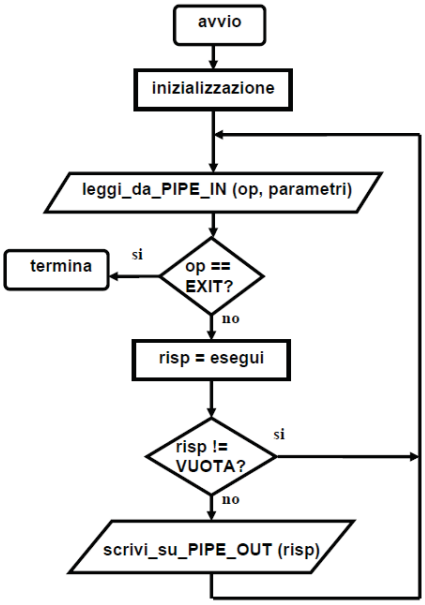


Figura 3.7: Diagramma di flusso del ciclo di vita del processo di riconoscimento.

classificatore con l'insieme dei campioni in appendice A.1. Questo tipo di operazione non ha parametri al seguito e non produce una risposta, vista la rapidità di esecuzione dell'allenamento non avrebbe infatti senso far eseguire all'interfaccia grafica le operazioni di lettura su PIPE;

2. deve essere eseguito l'allenamento con i campioni specificati dall'utente. In tal caso l'utente ha indicato i path dei file che vuole vengano usati come campioni. Anche in questo caso non viene fornita una risposta all'interfaccia;

3. è giunta la richiesta di creazione mappa, in questo caso i parametri devono contenere le coordinate relative alla latitudine e longitudine, la sua estensione, un nome e le note. Questi due ultimi parametri se non inseriti dall'utente sono comunque inizializzati con le stringhe "`new map`" e "`no note`", rispettivamente per il nome e per le note;

4. è giunta la richiesta di visionare una zona della mappa creata. I parametri devono quindi contenere le coordinate della zona da estrarre. Dall'interfaccia grafica giunge esclusivamente la richiesta dell'intera cartina, ma si potrebbe comunque richiedere un'estensione più piccola. Il

valore di ritorno dall'esecuzione dell'operazione 4 è una stringa contenente tutte le informazioni riguardanti l'area richiesta, in questo caso la mappa, che servono per la ricostruzione dell'immagine nel lato client. Per semplificare il passaggio dei parametri la mappa viene salvata come immagine .JPG e indicato al client il path dell'immagine;

5. giunge la richiesta di analisi. L'utente indica in quale directory sono state raccolte le immagini che devono essere analizzate per il riconoscimento della *Posidonia oceanica*. Una volta terminata la scansione della directory e la classificazione sulle immagini, il processo di riconoscimento indica la terminazione di questa operazione all'interfaccia grafica inserendo, sulla PIPE per le risposte, il valore dell'operazione appena eseguita. Una volta letto questo valore, l'interfaccia grafica che era rimasta in attesa, può sbloccarsi e l'utente può proseguire con altre operazioni;

6. giunge la richiesta di creare i campioni a partire dalle immagini contenute nella directory indicata dall'utente e passata come parametro dell'operazione. In questo caso il processo suddivide le immagini contenute nella directory in finestre di 150×150 e li salva nella directory `samples` che crea all'interno della directory passatagli dall'utente. Questo indirizzo sarà il valore di ritorno all'interfaccia grafica per consentirle di recuperare i campioni creati e farli visionare per la validazione dall'utente;

7. arriva la richiesta di apertura di un file `.xml`. Il file deve essere un file conforme al formato esposto precedentemente, di radice `<opencv_storage >` e contenere le informazioni per ricostruire il classificatore e la mappa;

8. arriva la richiesta di salvare i dati sul file `.xml` che è stato indicato nei parametri.

Due fasi di classificazione

Questa implementazione del sistema fornisce la predizione della classificazione in un tempo medio stimato di circa 77 secondi. Per raffinare l'accuratezza del risultato però è stato aggiunto al modello della Posidonia un secondo template per rappresentare le praterie in condizioni di maggiore oscurità, o profondità, e di conseguenza un secondo classificatore che, per la condizione

in cui si ricavi una estensione ritenuta sufficiente di *Posidonia oceanica* con la prima classificazione, raffina il risultato aggiungendo le zone più in ombra non riconosciute con il confronto dal primo template.

Questa soluzione ha portato ad un appesantimento dei calcoli computazionali fino a raggiungere un tempo medio di circa 150 secondi per il riconoscimento su una immagine di 264×198 pixel.

Questa scelta nella implementazione è stata imposta dalla impossibilità di tenere di conto della profondità alla quale viene rilevata l'immagine, ma sarebbe opportuno personalizzare il riconoscimento variando il template e l'allenamento del classificatore sulla base della informazione profondità. In questo modo si riuscirebbe a riportare i costi computazionali a livelli inferiori.

CAPITOLO 4

Validazione e Risultati

È stato formulato un modello di riconoscimento per la *Posidonia oceanica* ed implementata la relativa tecnica di classificazione. Il modello proposto deve essere validato per confermare la sua efficacia.

La validazione è stata effettuata sfruttando la tecnica nota come cross-validation, ripetendo il procedimento per due volte e analizzando i risultati ottenuti dal confronto tra le due fasi di validazione.

4.1 $K \times 2$ cross-validation

Il modello e la classificazione sono stati testati su un insieme di 48 immagini di 150×150 pixel che rappresentano esclusivamente *Posidonia oceanica* a diversi livelli di scala e con condizioni di limpidità dell'acqua diverse. L'insieme dei campioni sono riportati in appendice A.1.

Tale insieme è stato diviso in $K = 8$ gruppi disgiunti di foto e su ogni gruppo è stato testato il sistema sviluppato impostando come insieme per l'allenamento del classificatore l'unione degli altri $K - 1 = 7$ gruppi non usati per il test. Questo procedimento è stato ripetuto due volte dividendo in modo casuale le immagini sugli 8 gruppi e assicurandosi di non ripetere gli stessi insiemi proposti nella prima fase di validazione.

L'insieme complessivo dei 48 campioni di appendice A.1 è stato poi sottoposto a operazioni di thresholding, con vari valori di soglia, per consentire

di separare all'interno delle immagini, la Posidonia dal fondale. Su queste immagini l'operazione di thresholding ha funzionato egregiamente vista l'assenza di altre tipologie di oggetti diversi dalla Posidonia e la natura sabbiosa del fondale. Si è ottenuto così un insieme che pur non essendo un riconoscimento esatto della Posidonia, fornisce comunque una buona base di paragone per la valutazione degli errori di classificazione. Questo insieme, usato per la validazione, è visionabile in appendice A.2.

Le immagini restituite dalla classificazione sono riportate in appendice A.3 e A.4, rispettivamente per il primo ciclo di classificazione e per il secondo ciclo di classificazione. Questi risultati sono stati confrontati con l'insieme in appendice A.2 e per ogni immagine è stata calcolata la similarità tra la stessa immagine dall'insieme in appendice A.2 prima con l'immagine dall'insieme in appendice A.3 e poi con la corrispondente immagine in appendice A.4.

Ad una similarità maggiore corrisponde una maggiore *accuratezza* di classificazione, ossia definiamo l'accuratezza come la proporzione, in percentuale, tra l'area totale che doveva essere riconosciuta e quella che è stata effettivamente riconosciuta dal classificatore. Mentre un altro termine di valutazione è la *correttezza*. Per correttezza si intende infatti la probabilità che l'area che viene riconosciuta, contenga effettivamente la *Posidonia oceanica*.

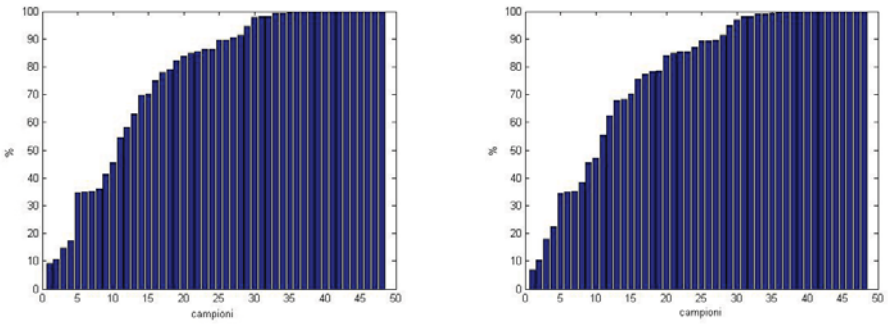
Quello che ci interessa ottenere è un'alta probabilità di correttezza, quindi che il classificatore operi con un tasso di errore il più basso possibile, ma non ci interessa una particolare accuratezza nel risultato; per il momento ci interessa di più individuare la zona, non l'estensione. Per fornire una valutazione di questi parametri sono state effettuate una serie di sperimentazioni ed osservandone i risultati potremmo ritenerci soddisfatti se la classificazione raggiungesse un tasso di accuratezza del 60% al quale, dalle sperimentazioni, è risultato corrispondere una correttezza di classificazione non inferiore all'80%.

Il confronto tra l'insieme di appendice A.2 e i due insiemi di classificazione è stato effettuato calcolando il "mean square error" tra gli istogrammi delle immagini. In Tabella 4.1 vengono mostrate le percentuali di similarità per ogni campione, tra gli insiemi delle sezioni di appendice A.2 e A.3 nella prima colonna, con l'appendice A.4 nella seconda colonna e nella terza sono riportate le medie delle due percentuali.

La colonna della media esprime la percentuale di accuratezza che mediamente si è ottenuta su ogni campione dalla operazione di classificazione con

I^ociclo	II^ociclo	media	I^ociclo	II^ociclo	media
99.90%	99.88%	99.89%	70.36%	70.29%	70.32%
100%	100%	100%	82.43%	77.38%	79.91%
14.78%	17.86%	16.32%	9.28%	6.99%	8.14%
85.04%	84.93%	84.98%	10.70%	10.30%	10.50%
79.18%	78.50%	78.84%	83.91%	84.14%	84.03%
45.53%	45.42%	45.48%	86.38%	85.48%	85.93%
99.25%	98.83%	99.04%	100%	100%	100%
91.31%	91.31%	91.31%	78.06%	78.19%	78.13%
100%	100%	100%	100%	100%	100%
89.56%	89.46%	89.51%	36.13%	35.37%	35.75%
98.19%	98.13%	98.16%	90.43%	89.75%	90.09%
35.38%	34.80%	35.09%	99.58%	99.60%	99.58%
34.97%	38.44%	36.70%	100%	100%	100%
100%	100%	100%	85.57%	87.09%	86.33%
100%	100%	100%	99.73%	99.26%	99.50%
86.48%	85.44%	85.96%	54.63%	55.42%	55.3%
99.38%	99.01%	99.19%	99.96%	100%	99.98%
100%	100%	100%	74.99%	75.58%	75.28%
34.76%	34.24%	34.50%	94.54%	94.78%	94.66%
89.49%	89.42%	89.45%	99.76%	99.88%	99.82%
17.26%	22.49%	19.88%	69.72%	68.41%	69.07%
58.2%	68%	63.10%	98.22%	98.20%	98.21%
63.14%	62.57%	62.86%	41.45%	47.18%	44.32%
97.85%	96.99%	97.42%	100%	100%	100%

Tabella 4.1: Percentuali di accuratezza delle classificazioni.



(a) Primo ciclo di validazione. (b) Secondo ciclo di validazione.

Figura 4.1: Istogrammi delle percentuali di Tabella 4.1

il sistema implementato.

Osservando anche gli istogrammi di Fig. 4.1 si può notare che più della metà dei campioni hanno avuto un riconosciemento più che soddisfacente: se calcoliamo il valore mediano dei risultati si ottiene infatti che il sistema classifica la presenza di Posidonia con una accuratezza di circa il 90% sulla metà dei campioni.

Possiamo vedere le distribuzioni dei campioni sulle percentuali di accuratezza dal grafico di Fig. 4.2.

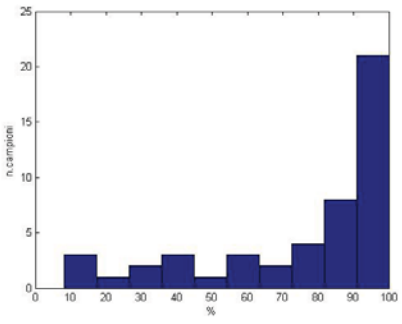


Figura 4.2: Distribuzione dei campioni sulle percentuali di accuratezza ottenuta.

Osservando la Tabella 4.1 notiamo che solo 10 campioni del nostro insieme in appendice A.1 hanno un livello di accuratezza inferiore al 50% (riportati in Tabella 4.2); se confrontiamo le classificazioni fatte dal sistema con l'immagine originale possiamo ritenere non tutti i campioni di Tabella 4.2 errati. Infatti la classificazione sui campioni (i) e (j) può essere ritenuta comple-

tamente corretta. Risulta corretta anche la classificazione su (b), (c) e (d) anche se con accuratezza minore e comunque non inferiore al 70%. Mentre si può sicuramente ritenere completamente errata la classificazione su (e). Come anche le classificazioni di (a), (f), (g) e (h) possono essere ritenute errate o con scarsissima accuratezza.

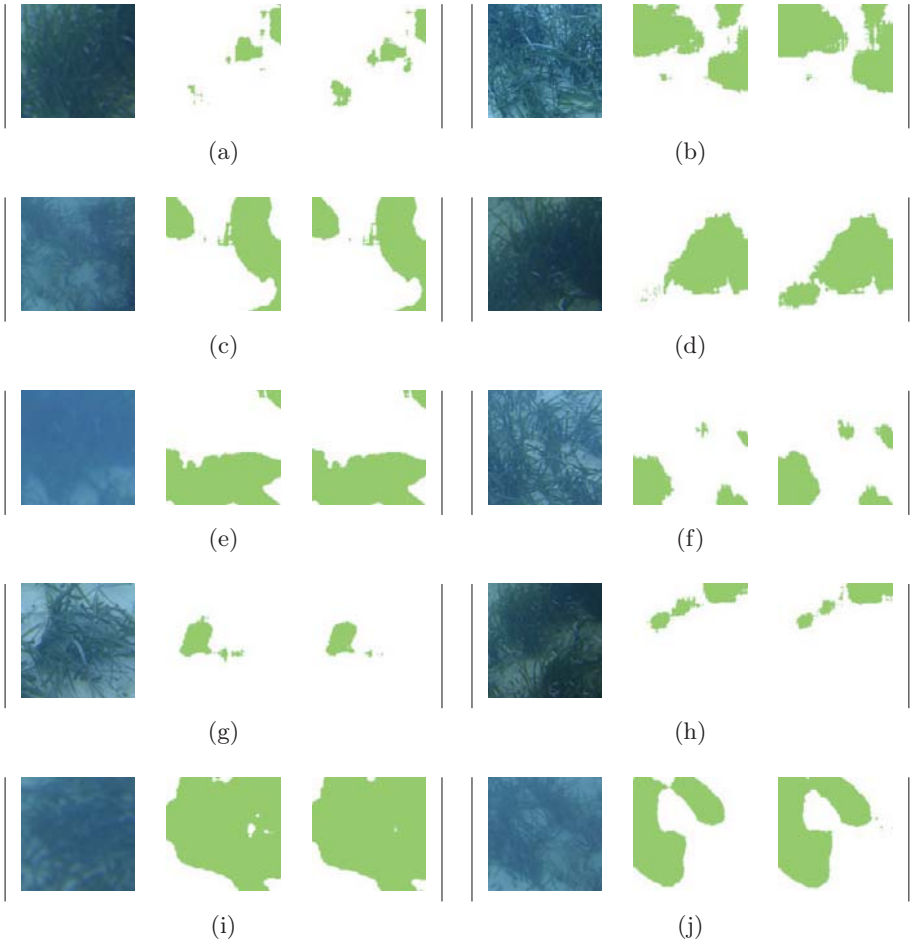


Tabella 4.2: Campioni con accuratezza $< 50\%$.

Fatte queste considerazioni e presupponendo che i campioni con accuratezza maggiore o uguale al 50% possano essere considerati classificati correttamente, allora deduciamo che il sistema esegue una classificazione con un tasso di errore pari al 10.42%. I valori ottenuti soddisfano quindi pienamente le nostre esigenze.

4.2 Discussione

Riguardo all'accuratezza ottenuta possiamo far notare, come già spiegato in Sezione 3.1, che la precisione del riconoscimento dipende strettamente dalle dimensioni dell'immagine template utilizzata nel modello. Da varie prove si era notato infatti che un template più grande produceva risultati con accuratezza minore fino ad arrivare ad una classificazione da potersi ritenere anche errata. Mentre un'immagine template eccessivamente piccola avrebbe apportato un maggior carico computazionale producendo risultati non buoni.

Inoltre, possiamo far notare che la maggior parte dei campioni raccolti sono stati prodotti in condizioni di scarsa luminosità. Ci si può chiedere perciò se in condizioni di luminosità diverse il sistema sia comunque in grado di riconoscere il pattern della Posidonia. La forza del modello proposto è proprio quella di poter personalizzare l'allenamento del classificatore in modo tale da poter riconoscere anche un pattern con luminosità diversa rispetto ai campioni di appendice A.1, limitandosi a modificare soltanto l'insieme di allenamento. Proponiamo un esempio di riconoscimento in Tabella 4.3 dove il classificatore produce i risultati ivi riportati in seguito ad un allenamento con i campioni raccolti in appendice A.5.

Tabella 4.3: Tre esempi di riconoscimento su campioni con alta luminosità.

Nonostante l'insieme di allenamento non abbia un'alta cardinalità, i riconoscimenti ottenuti sono da ritenersi soddisfacenti.

I risultati ottenuti da questo processo di validazione dimostrano anche che è stato ottenuto un buon equilibrio tra il numero dei campioni, il numero delle feature e la complessità del classificatore.

Su ogni immagine campione vengono estratti 50 vettori delle feature $\in \mathbb{R}^5$ e l'intero algoritmo di estrazione feature e classificazione, sulle immagini di 150×150, impiega mediamente 50 secondi.

La dimensionalità dei vettori delle feature passati al classificatore è piuttosto ridotta. Questo consente di mantenere i costi computazionali ragionevoli ma allo stesso tempo di raggiungere un alto tasso di correttezza.

Conclusioni

L'obiettivo di questa tesi era quello di ricercare e formulare, se esisteva, un modello di riconoscimento per la *Posidonia oceanica* su immagini fotografiche di fondali marini.

È stato dimostrato, nei capitoli precedenti, la possibilità di formulare tale modello ed è stata fornita una sua implementazione. Il sistema di riconoscimento è stato poi validato con una serie di campioni e i risultati hanno confermato la fattibilità della formulazione proposta.

Per determinare il modello di riconoscimento è stato adottato un approccio di tipo misto: partendo dall'idea del Template Matching si è costruito un modello strutturale del pattern Posidonia dal quale poi si è proceduto a determinare statisticamente il vettore delle feature per applicare una classificazione tramite l'impiego delle Support Vector Machine.

È stato prima di tutto necessario ricercare una immagine del pattern, qualificata come template; per questo compito sono state necessarie varie prove per fissare la giusta dimensione e scala.

Successivamente è stato selezionato un insieme di filtri di Gabor per creare l'insieme delle maschere di convoluzione con cui filtrare le immagini.

Sulle immagini filtrate sono stati poi calcolati gli istogrammi e da questi determinato il vettore delle feature come vettore delle distanze, calcolate secondo la statistica χ^2, tra finestre appartenenti all'immagine in fase di analisi e all'immagine template.

Il vettore delle feature così estratto viene quindi passato al classificatore per l'operazione di predizione.

Per questa operazione è stato impiegato un classificatore di tipo SVM, che determina la presenza o meno della Posidonia nella finestra presa in cosiderazione sull'immagine in fase di analisi, grazie ad un precedente allenamento con il quale ha potuto individuare l'iperpiano di separazione tra le due classi coinvolte: classe Posidonia e classe Non Posidonia.

Una volta formulato il modello di rappresentazione per il pattern Posidonia e stabilito il classificatore da usare, si è proceduto con l'implementazione del sistema di riconoscimento.

Verso questo sistema è stata anche fornita una interfaccia grafica minimale in modo da mostrare i risultati ottenuti e presentare una semplice applicazione della tecnica di riconoscimento proposta.

La soluzione fornita al problema del riconoscimento della *Posidonia oceanica*, con la sua implementazione, è stata formulata modularmente ed è perciò facilmente integrabile all'interno di diversi scenari applicativi.

Guardando alle motivazioni sulle quali si è fondato questo lavoro di ricerca, il sistema di riconoscimento sviluppato potrebbe essere già usato per la valutazione dei maggiori descrittori dello stato di conservazione delle praterie di *Posidonia oceanica*:

- ubicazione;
- estensione;
- profondità del limite superiore;
- profondità e tipologia del limite inferiore;
- densità della prateria.

La tecnica di riconoscimento implementata è capace di individuare le zone di interesse con una accuratezza di circa il 77% ed una correttezza di quasi il 90%. I risultati ottenuti dimostrano la possibilità di realizzare un sistema automatico per il riconoscimento della *Posidonia oceanica*, alternativo alle tecniche impiegate fino ad ora, capace di operare in condizioni anche di scarsa luminosità e visibilità. Anche se le condizioni ambientali risultano influenzare fortemente la sua abilità, il modello formulato può essere facilmente adattato alle esigenze del caso.

Per migliorare i risultati della classificazione, oltre a consentire all'utente di personalizzare l'allenamento del classificatore, si può anche pensare di

cambiare l'algoritmo di riconoscimento per adattare l'insieme dei campioni di allenamento, e probabilmente anche il template, in modo automatico al variare della profondità rilevata, della luminosità o della limpidità dell'acqua.

Potrebbe essere anche interessante andare a vedere cosa succede cambiando i parametri del classificatore SVM in modo da consentire un riconoscimento di classi multiple per esempio per estendere il riconoscimento anche al tipo di fondale (sabbioso o roccioso) o per il riconoscimento delle *mattes*, la particolare struttura costituita dai rizomi di sedimentazione dalla quale cresce la *Posidonia oceanica*.

Lavorare su questa tesi ha dato l'opportunità di allargare le proprie conoscenze verso il mondo della Computer Vision e il Machine Learning, in particolar modo nel riconoscimento e nella classificazione su immagini. Si sono scoperte le tecniche più diffuse nell'elaborazione di immagini e gli strumenti più utili per la risoluzione di questo tipo di problemi.

Il lavoro è stato svolto non senza difficoltà ma apprezzando in ogni caso il fascino della materia e riuscendo a formulare un modello ed un approccio per la risoluzione del problema di Pattern Recognition specifico.

Partendo dall'analisi del problema, formulando un approccio risolutivo fino ad arrivare all'implementazione della tecnica di riconoscimento, si è avuta l'opportunità di lavorare su di un progetto per il suo intero ciclo di vita imparando così a distinguerne le varie fasi ed affrontarne le singole problematiche passo per passo.

APPENDICE A

Immagini

A.1 Campioni

In Tabella A.1 sono raccolti i 48 campioni di dimensione 150×150 pixel rappresentanti la *Posidonia oceanica* usati nella validazione del sistema di riconoscimento implementato.

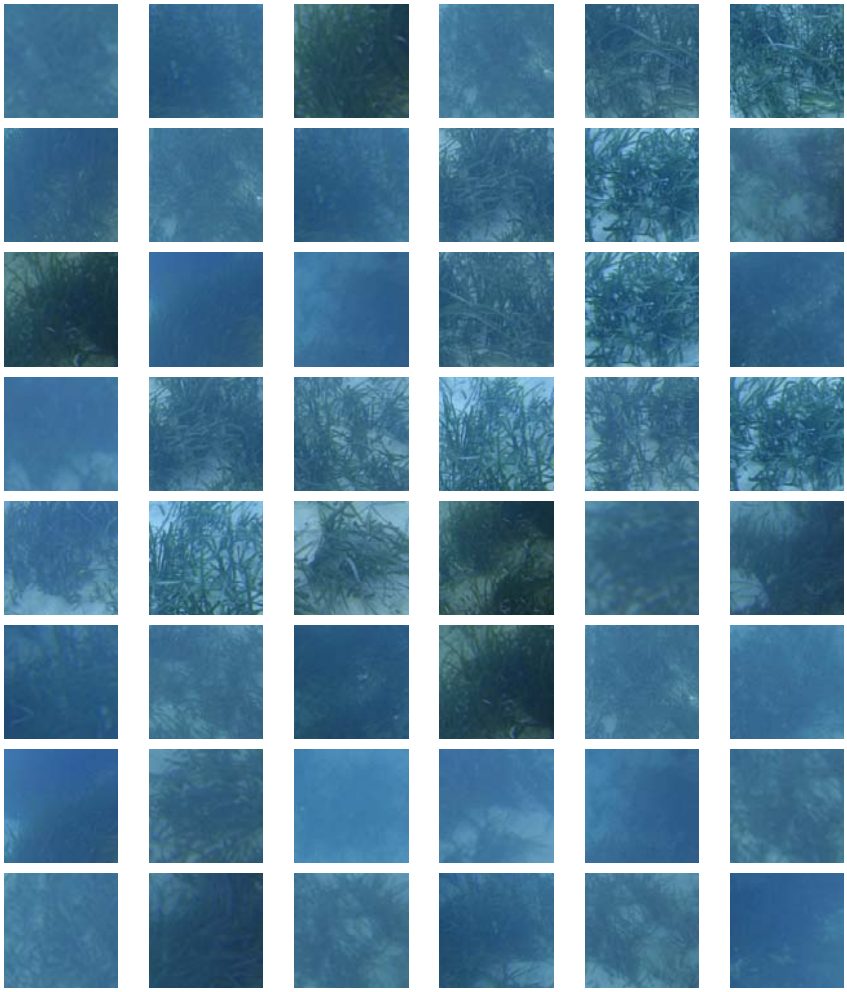


Tabella A.1: Insieme dei campioni.

A.2 Insieme di validazione

I campioni di Tabella A.1 sono stati sottoposti a operazioni di thresholding tramite `cvThreshold(src,dst,soglia,255,CV_THRESH_BINARY)` diversificando i valori di `soglia` per i campioni con più o meno luminosità e con livelli di scala diversi. I risultati sono raccolti in Tabella A.2; le zone nere sono quelle con valori minori alla soglia, quindi riconducibili alla Posidonia mentre le bianche indicano le zone del fondale.

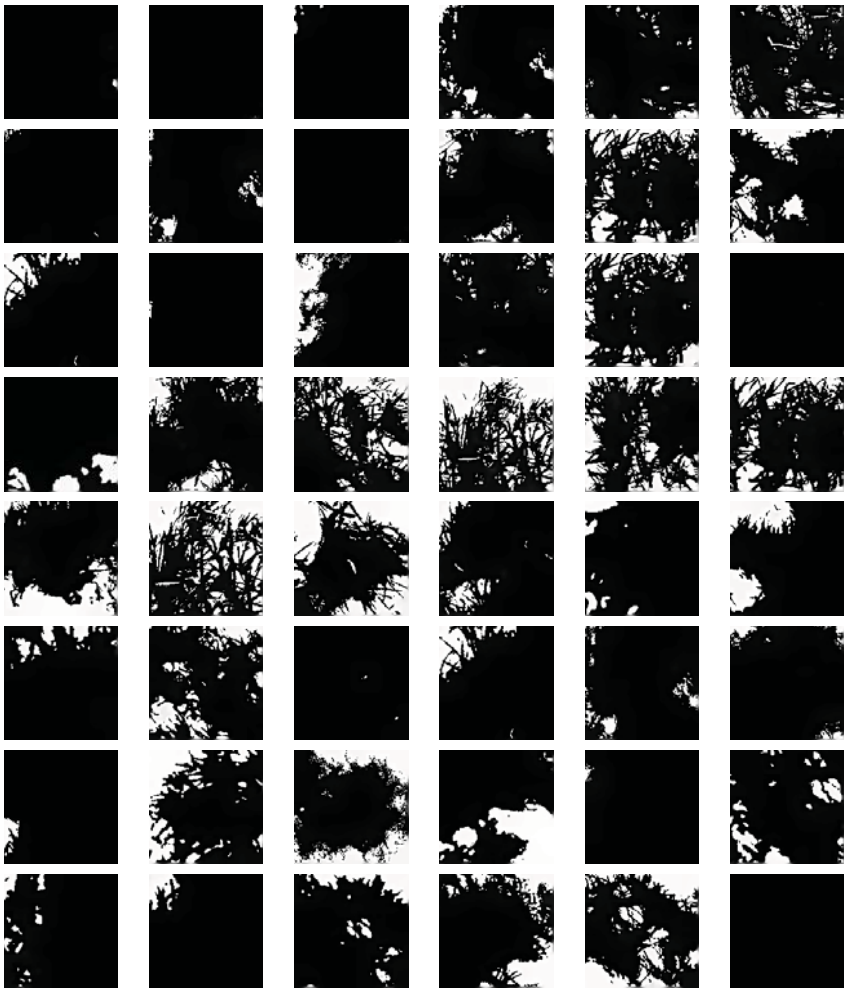


Tabella A.2: Risultato delle operazioni di thresholding sui campioni.

A.3 Prima validazione

In Tabella A.3 sono raccolti i risultati della classificazione dei campioni di Tabella A.1. Le zone verdi rappresentano la presenza di Posidonia mentre nelle zone bianche è stata classificata l'assenza.

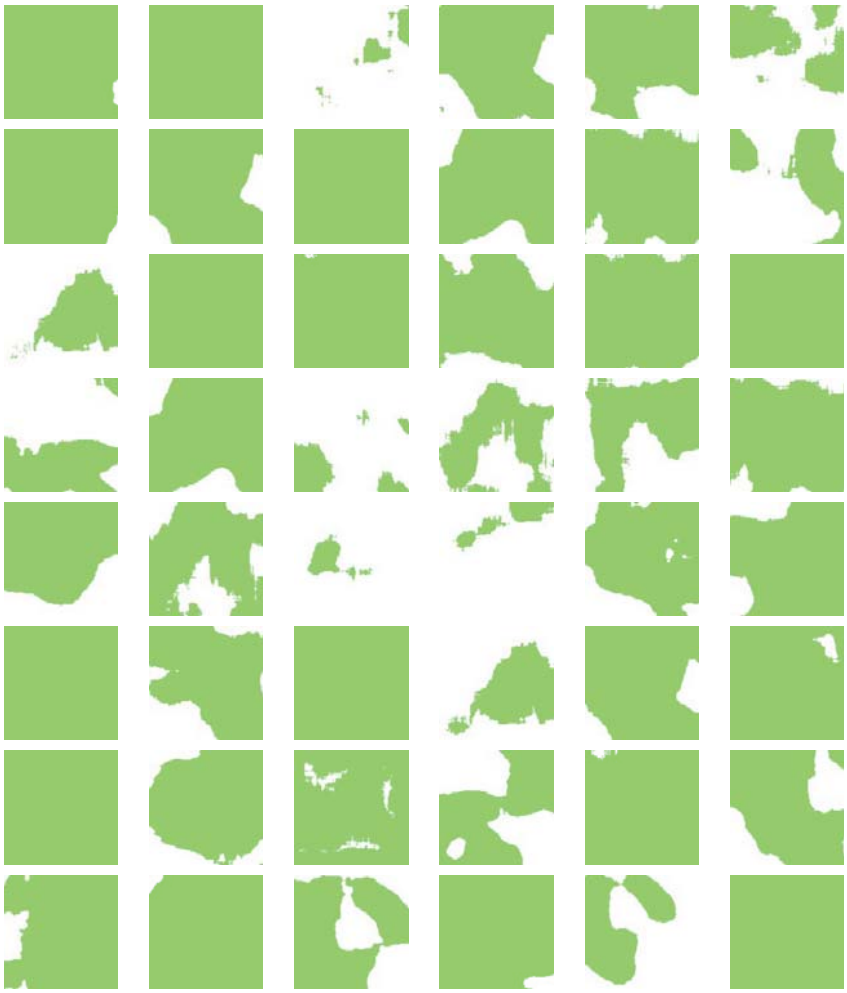


Tabella A.3: Risultato della classificazione con il primo ciclo di validazione.

A.4 Seconda validazione

In Tabella A.4 sono raccolti i risultati del secondo ciclo di validazione sui campioni di Tabella A.1. Come per la sezione precedente, il verde indica la presenza di Posidonia e il bianco l'assenza.

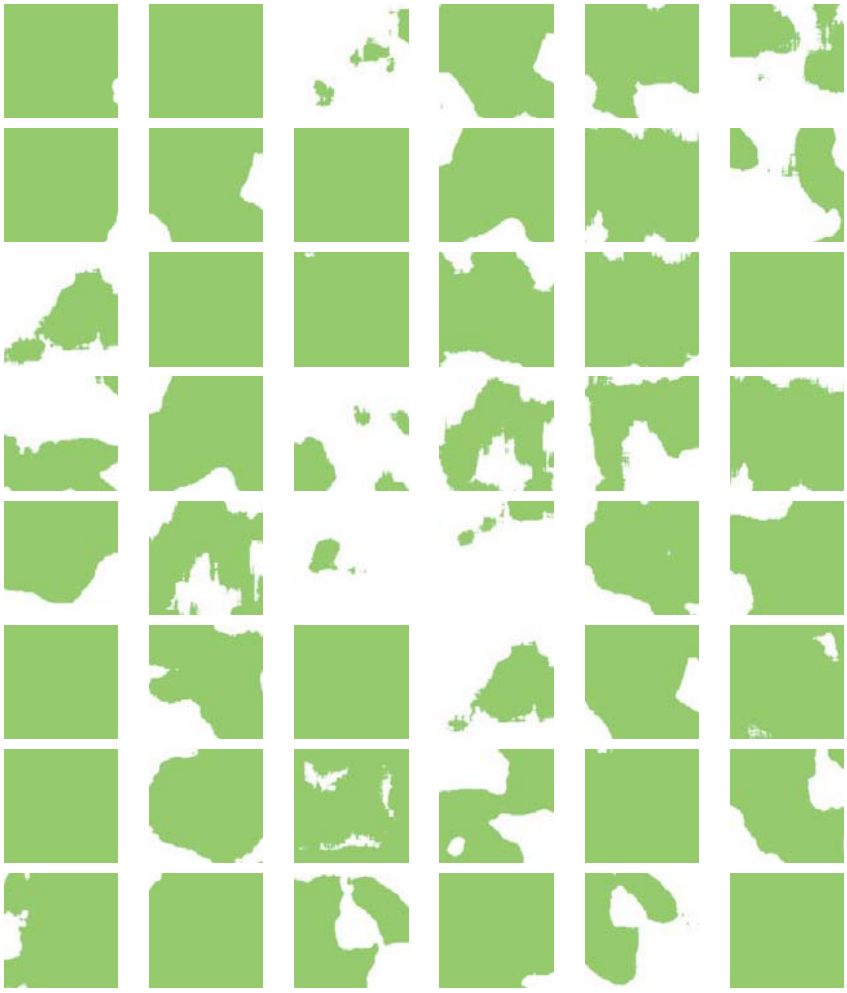


Tabella A.4: Risultato della classificazione con il secondo ciclo di validazione.

A.5 Seconda campionatura

In Tabella A.5 è mostrato l'insieme dei campioni usati per l'allenamento del classificatore per il riconoscimento di *Posidonia oceanica* in condizioni di luminosità elevata. Campionati per la validazione del modello di riconoscimento.

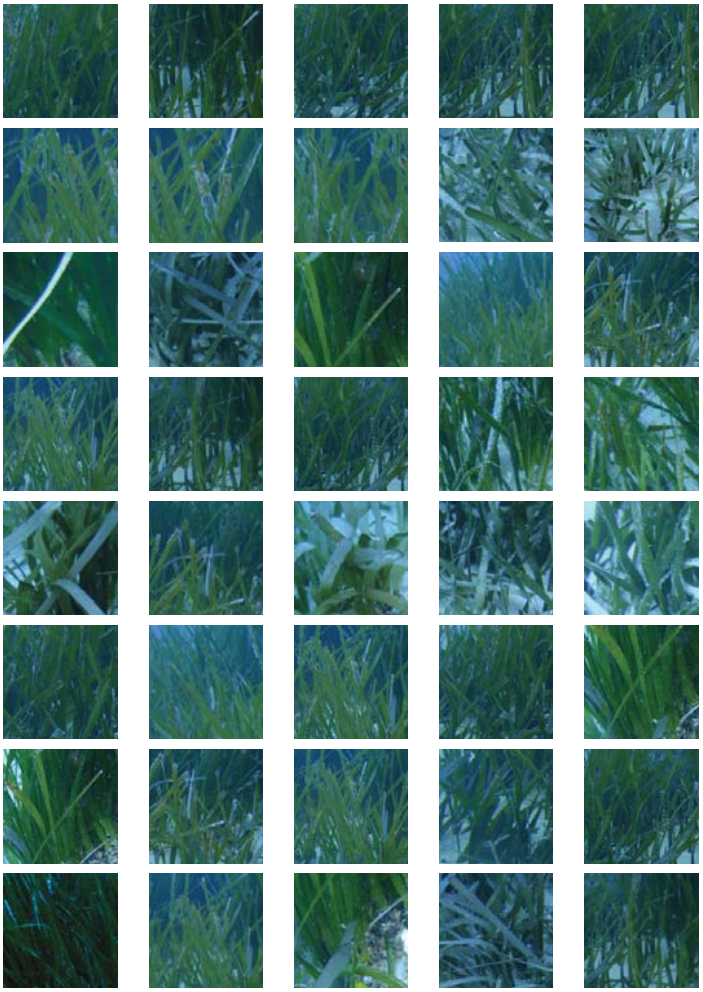


Tabella A.5: Insieme dei campioni con alta luminosità.

APPENDICE B

Documentazione

Per il modulo di riconoscimento è stata prodotta la documentazione delle classi, con l'ausilio di Doxygen[1] versione 1.7.1, di seguito riportata.

B.1 PosidoniaModel Class Reference

Public Member Functions

- **PosidoniaModel** ()

  *Costruttore per l'oggetto **PosidoniaModel** (p. 74).*

- **PosidoniaModel** (const **PosidoniaModel** &orig)
- **PosidoniaModel** (char *templfile)

  *Costruttore per l'oggetto **PosidoniaModel** (p. 74) per il quale si specifica il template da usare.*

- void **initModel** ()

  Funzione di inizializzazione del modello. Deve essere obbligatoriamente invocato subito dopo il costruttore, altrimenti falliscono tutte le operazioni.

- IplImage * **getPosidoniaTemplate** ()

[1] http://www.stack.nl/~dimitri/doxygen/index.html

Restituisce l'oggetto immagine del template usato.

- void **filterImage** (IplImage ∗src, IplImage ∗dst, int filter)
- double **getHistDistance** (int templ, IplImage ∗img, int filter)
- void **trainAuto** ()

  Effettua il train automatico, selezionando come campioni per l'allenamento un insieme predefinito di immagini.

- void **train** (vector< IplImage ∗ >)

  Effettua l'allenamento del classificatore sull'insieme di immagini passategli come parametro.

- void **predictPos** (IplImage ∗src, IplImage ∗dst)
- void **buildTrainingData** (vector< IplImage ∗ >, CvMat ∗, int)

  Costruisce il vettore delle feature sull'insieme di immagini passate.

- void **buildTrainingData** (string ∗, int, CvMat ∗, int)

  Costruisce il vettore delle feature sull'insieme di immagini ritrovate agli indirizzi passati.

- CvSVM ∗ **getClassifier** (int)

  Ritorna il classificatore. Accetta come parametro un flag per indicare il classificatore principale oppure il secondario.

- void **setClassifier** (int, CvSVM ∗)

  Imposta come classificatore quello passatogli come parametro.

B.1.1 Member Function Documentation

B.1.1.1 void PosidoniaModel::filterImage (IplImage ∗ *src*, IplImage ∗ *dst*, int *filter*)

Esegue l'operazione di filtraggio.

Parameters

src - immagine sorgente su cui eseguire il filtraggio

dst - immagine destinazione, risultante dal filtraggio

filter - filtro da eseguire su src

B.1.1.2 double PosidoniaModel::getHistDistance (int *templ,* IplImage * *img,* int *filter*)

Restituisce la distanza chi-quadro tra gli istogrammi.

Parameters

templ - template con il quale fare il confronto; puo' essere 1 per template principale, 0 per secondario;

img - e' l'immagine sulla quale occorre calcolare la distanza

filter - e' il filtro da applicare e confrontare

Returns

la distanza degli istogrammi tra templ e img filtrati

B.1.1.3 void PosidoniaModel::predictPos (IplImage * *src,* IplImage * *dst*)

Effettua la predizione del classificatore.

Parameters

src - immagine sorgente sulla quale effetture la predizione

dst - mette qui l'immagine con i valori di predizione. Nelle zone senza Posidonia mette DATA_EMPTY, dove trova Posidonia mette DATA_POS

B.2 TrackMap Class Reference

Public Member Functions

- **TrackMap** ()

  Costruttore per l'oggetto ***TrackMap*** *(*p. *76**).*

- **TrackMap** (float nw_x, float nw_y, float se_x, float se_y)
- int **add** (IplImage *img, float nw_x, float nw_y)
- IplImage * **get** (float nw_x, float nw_y, float se_x, float se_y)

- void **reset** ()

  Azzera il contenuto della cartina.

- void **setNote** (const char *s)

  Setta il valore delle note.

- void **setName** (const char *s)

  Setta il valore del nome della cartina.

- void **setScale** (float s)

  Setta la scala della cartina.

- void **setCoord** (float nw_x, float nw_y, float se_x, float se_y)
- void **setCell** (string key, IplImage *item)
- char * **getNote** ()

  Restituisce le note.

- char * **getName** ()

  Restituisce il nome.

- float **getScale** ()

  Restituisce il valore della scala.

- float **getNorthWest_X** ()

  Restituisce la x dell'estremo a Nord-Ovest.

- float **getNorthWest_Y** ()

  Restituisce la y dell'estremo a Nord-Ovest.

- float **getSouthEast_X** ()

  Restituisce la x dell'estremo a Sud-Est.

- float **getSouthEast_Y** ()

  Restituisce la y dell'estremo a Sud-Est.

- subMap::iterator **begin** ()

  Restituisce l'iteratore sulle celle della cartina a partire dal primo elemento.

- subMap::iterator **end** ()

  Restituisce l'iteratore sulle celle della cartina a partire dall'ultimo elemento.

B.2.1 Constructor & Destructor Documentation

B.2.1.1 TrackMap::TrackMap (float *nw_x*, float *nw_y*, float *se_x*, float *se_y*)

Costruttore per l'oggetto **TrackMap** (p. 76). Accetta gli estremi delle coordinate.

Parameters

nw_x - coordinata x nord-ovest

nw_y - coordinata y nord-ovest

se_x - coordinata x sud-est

se_y - coordinata y sud-est

B.2.2 Member Function Documentation

B.2.2.1 int TrackMap::add (IplImage * *img*, float *nw_x*, float *nw_y*)

Aggiunge una immagine nella cartina.

Parameters

img - l'immagine da aggiungere

nw_x - coordinata della immagine, x a nord-ovest (latitudine)

nw_y - coordinata della immagine, y a nord-ovest (longitudine)

Returns

1 se l'operazione va a buon fine, 0 altrimenti

B.2.2.2 IplImage * TrackMap::get (float *nw_x*, float *nw_y*, float *se_x*, float *se_y*)

Recupera una zona dalla cartina.

Parameters

nw_x - coordinata x a nord-ovest (latitudine) che vogliamo recuperare

nw_y - coordinata y a nord-ovest (longitudine) che vogliamo recuperare

se_x - coordinata x a sud-est (latitudine) che vogliamo recuperare

se_y - coordinata y a sud-est (longitudine) che vogliamo recuperare

Returns

l'immagine della zona richiesta

B.2.2.3 void TrackMap::setCell (string *key*, IplImage * *item*)

Inserisce una cella intera nella cartina.

Parameters

key - chiave corrispondente all'immagine da inserire, coordinate della cella

item - immagine della cella da inserire

B.2.2.4 void TrackMap::setCoord (float *nw_x*, float *nw_y*, float *se_x*, float *se_y*)

Setta le coordinate della cartina.

Parameters

nw_x - coordinata x a nord-ovest (latitudine)

nw_y - coordinata y a nord-ovest (longitudine)

se_x - coordinata x a sud-est (latitudine)

se_y - coordinata y a sud-est (longitudine)

B.3 ImgGeoTIFF Struct Reference

`#include <utilities.h>`

Public Attributes

- IplImage ∗ **image**
- string ∗ **filename**
- float **latitude**
- float **longitude**

B.3.1 Detailed Description

Struttura per raccogliere le informazioni delle immagini georeferenziate. Aggiungere qui le informazioni riguardanti la profondita' e la luminosita': float depth; float light;

Bibliografia

[1] APAT/RAPPORTI. Tutela della connetività ecologica degli habitat marini e costieri: una proposta per l'organizzazione e la gestione dei dati. Technical report, APAT - Agenzia per la protezione dell'ambiente e per i servizi tecnici, 2005.

[2] G. Ferri, F. Fornai, A. Manzi, G. Santerini, F. Ciuchi, A.J. Lee, C. Laschi, B. Mazzolai, and P. Dario. HydroBot: un catamarano autonomo per monitoraggio ambientale. SEA-MED, 2010.

[3] A.K. Jain, Robert P.W. Duin, and J. Mao. Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Learning*, 22(1):4–37, Gennaio 2000.

[4] D. Devescovi. Impiego di classificatori nell'analisi di immagini. `http://www.devedeve.com/wp-includes/files/Impiego%20di%20Classificatori.pdf`.

[5] Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, (2):121–167, 1998.

[6] S. Haykin. *Neural Networks, A Comprehensive Foundation.* Prentice Hall, 1999.

[7] B. Schölkopf, A. Smola, and K.R. Muller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. In *Neural Computation*, volume 10, pages 1299–1319. 1998.

[8] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[9] V. Vapnik, S. Golewich, and A. Smola. Support vector method for function approximation, regression estimation and signal processing. In *Advances in Neural Information Processing Systems*, pages 281–287. 1996.

[10] M. Tuceryan and A.K. Jain. Texture Analysis. In *Handbook of Pattern Recognition and Computer Vision*, pages 235–276. World Scientific, 1992.

[11] H. Tamura, S. Mori, and Y. Yamawaki. Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-8:460–473, Giugno 1978.

[12] J. Sklansky. Image segmentation and feature extraction. *IEEE Transactions on Systems, Man and Cybernetics*, pages 237–247, 1978.

[13] J.K. Hawkins. Textural properties for pattern recognition. In *Picture Processing and Psychopictorics*. Academic Press, 1969.

[14] B. Julesz, E. N. Gilbert, L. A. Shepp, and H. L. Frisch. Inability of humans to discriminate between visual textures that agree in second-order statistics. In *Perception*, volume 2, pages 391–405. 1973.

[15] B. Julesz. Visual pattern discrimination. In *IRE Transactions on Information Theory*, volume 8, pages 84–92. 1962.

[16] B. Julesz. Experiments in the visual perception of texture. In *Scientific America*, volume 232, pages 34–43. 1975.

[17] B. Julesz. Nonlinear and cooperative processes in texture perception. In *Theoretical Approaches in Neurobiology*, pages 93–108. MIT Press, 1981.

[18] B. Julesz. Textons, the elements of texture perception, and their interactions. In *Nature*, pages 91–97. 1981.

[19] S.C. Zhu, C. Guo, Y. Wang, and Z. Xu. What Are Textons? *International Journal of Computer Vision*, 62(1/2):121–143, 2005.

[20] P. Zamperoni. *Metodi dell'elaborazione digitale di immagini.* Masson.

[21] G. Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. In *Deuxième mémoire: Recherches sur les parallélloèdres primitifs*, volume 134, pages 198–287. J. Reine Angew. Mathematik, 1908.

[22] R. Chellappa and S. Chatterjee. Classification of Texture using Gaussian Markov random fields. *IEEE Transactions Acoustics Speech and Signal Processing*, 33:959–963, 1985.

[23] A. Khotanzad and R. Kashyap. Feature Selection for texture recognition based on image synthesis. *IEEE Transactions on Systems, Man and Cybernetics*, 17:1087–1095, 1987.

[24] Z. Tu and S.C. Zhu. Image Segmentation by Data-Driven Markov Chain Monte Carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):657–673, Maggio 2002.

[25] G. C. Cross and A. K. Jain. Markov random field texture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (5):25–39, 1983.

[26] X. Liu and D. Wang. Texture Classification using Spectral Histograms. *IEEE Transactions on Image Processing*, 12(6):661–670, Giugno 2003.

[27] M. Varma and A. Zisserman. A Statistical Approach to Texture Classification from Single Images. *International Journal of Computer Vision*, 62(1/2):61–81, 2005.

[28] Z. Tu, X. Chen, A. L. Yuille, and S.C. Zhu. Image Parsing: Unifying Segmentation, Detection and Recognition. *International Journal of Computer Vision*, 63(2):113–140, 2005.

[29] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the 13th International Conference*, 1996.

[30] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. Technical report, Department of Statistics, Stanford University, 1998.

[31] P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In *Advances in Neural Information Processing System*, pages 1311–1318. MIT Press, 2001.

[32] X. Pan and Q.Q. Ruan. Palmprint recognition using Gabor feature-based $(2D)^2$PCA. *Neurocomputing*, (71):3032–3036, 2008.

[33] G. Bradski and A. Kaehler. *Learning OpenCV, Computer Vision with the OpenCV Library.* O'Reilly, 2008.