ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA
SEDE DI CESENA
FACOLTA' DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea in Scienze e Tecnologie Informatiche

UN SISTEMA AUTOMATICO PER IL COLLAUDO DI BOMBOLE A GAS

Relazione finale in

Tecniche di Elaborazione di Immagini

Relatore:

Prof. Damiana Lazzaro

Presentata da:

Davide Solazzi

Sessione II

Anno Accademico 2011/2012

Indice

Elenco delle figure

Introduzione

Lo scopo di questa tesi è di sviluppare un'applicazione in grado di effettuare il riconoscimento automatico della data di prossimo collaudo riportata sulle etichette di bombole a gas. L'utilizzo di un sistema di questo tipo consente da un lato l'aumento del numero di collaudi, in quanto effettuati da un macchinario completamente automatizzato che lavora *real time*, mentre dall'altro permette di risparmiare nell'impiego di tecnici autorizzati a effettuare questo tipo di accertamenti. L'applicazione riceve in ingresso una immagine acquisita da un sensore, che rappresenta la parte superiore della bombola contenente l'etichetta, ed attraverso gli algoritmi sviluppati fornisce in uscita la data riconosciuta. Nel caso in cui la data di collaudo superi quella attuale la bombola dovrà essere nuovamente collaudata. Questi sistemi di riconoscimento automatico hanno avuto una grande diffusione soprattutto negli ultimi anni. Essi creano nuovi scenari applicativi sia a livello accademico che industriale. Basti pensare, per esempio, ai vantaggi che porterebbe un sistema di video sorveglianza automatizzato in grado di effettuare il riconoscimento di bagagli incustoditi in aeroporti o ferrovie. Un'altra possibile applicazione è data dal riconoscimento di volti che consente di aumentare notevolmente il livello di sicurezza sia in ambito pubblico che aziendale. Da questi esempi si può capire come l'uso di tecniche di riconoscimento automatico possa portare numerose innovazioni tecnologiche in tutti i campi, dal settore sociale a quello scientifico o aziendale. Nel nostro caso è stato sviluppato un sistema che porta importanti vantaggi in applicazioni industriali consentendo un aumento della produzione, ma anche un maggior risparmio sui costi legati al personale.

La realizzazione dell'applicazione in grado di effettuare il riconoscimento della data di collaudo è stata effettuata utilizzando tre sottofasi principali: come prima operazione si individua l'etichetta sopra la quale è impresso l'anno, utilizzando

una tecnica di *tresholding*, in cui come valore di soglia si sfrutta una determinata intensità luminosa dei pixel, ottenuta in modo empirico. Nella seconda sottofase si procede alla localizzazione del pittogramma, utilizzando nelle immagini in cui è possibile la stessa tecnica usata nella precedente fase, altrimenti individuandone i lati sfruttando la trasformata di Hough. Una volta identificato il marchio ci si sposterà in posizione diametralmente opposta per collocarsi sulla data di prossimo collaudo. L'operazione conclusiva di classificazione viene realizzata utilizzando tecniche matematiche, detti momenti invarianti di Hu, per il riconoscimento dei numeri che costituiscono l'anno anche in caso di eventuali traslazioni, rotazioni o scalature dell'immagine. Per tutti i dettagli relativi all'implementazione si rimanda ai successivi capitoli.

Il *software* è stato realizzato in MatLab (abbreviazione di *Matrix Laboratory*), che risulta essere un ottimo ambiente di sviluppo per la costruzione di prototipi di applicazioni per l'elaborazione delle immagini.

La tesi è così strutturata: nel Capitolo 1 viene prima effettuata un'analisi dettagliata del problema affrontato. In seguito vengono fornite le nozioni teoriche alla base del *pattern recognition* ed una classificazione di tali sistemi. Nel Capitolo 2 si descrivono tutti i procedimenti che a partire dall'acquisizione di un'immagine la trasformano in una rappresentazione digitale su calcolatore. Successivamente vengono argomentate approfonditamente tutte le fasi che costituiscono un sistema di *pattern recognition*. Il Capitolo 3 è riservato alla descrizione dell'applicazione, in cui si trattano tutti i dettagli tecnici legati alla sua implementazione. Infine nel Capitolo 4 viene effettuata un'analisi dei tempi di esecuzione degli algoritmi implementati e vengono trattati i principali problemi delle immagini riscontrati durante la classificazione.

Capitolo 1

Pattern Recognition

In questo primo capitolo viene prima effettuata una breve descrizione del problema affrontato. Successivamente vengono introdotte le nozioni di base del *pattern recognition*, fondamentali per una corretta comprensione degli argomenti trattati nella tesi. Vengono poi classificati e descritti i vari sistemi esistenti. Per concludere viene analizzata una particolare applicazione di riconoscimento, detta *optical character recognition*.

1.1 Analisi del problema affrontato

Il progetto oggetto di questa tesi riguarda proprio un problema di *optical character recognition*. L'obiettivo da raggiungere consiste nello sviluppo di un sistema software in grado di riconoscere la data di prossimo collaudo dell'etichette delle bombole di gas. La soluzione di tale problema tramite tecniche di riconoscimento automatico può portare notevoli vantaggi, basti pensare al tempo che impiegherebbe un umano per effettuare l'identificazione e al tempo che impiegherebbe una macchina opportunamente addestrata per effettuare controlli *real time*. L'utilizzo di queste tecniche nei processi industriali sta assumendo sempre più importanza con il passare degli anni, proprio perché si possono avere grandi vantaggi, come l'utilizzo di un controllo qualità automatizzato o l'incremento della produzione. Uno dei possibili problemi del riconoscimento automatico è dato dalla probabilità di effettuare una classificazione errata, ma grazie ai numerosi studi ed alle ricerche sono stati individuati algoritmi in grado di raggiungere un livello di precisione vicino al 100%. Il progetto da me svolto nasce da una esigenza reale di individuazione delle bombole la cui data di prossimo collaudo è superiore alla data attuale.

Figura 1.1 Esempio di immagine su cui viene effettuato il riconoscimento

Le bombole tramite un nastro trasportatore passano sotto un dispositivo di acquisizione che effettua una foto della parte superiore contenente l'etichetta della bombola (vedi Figura 1.1). Il sensore che si occupa di acquisire l'immagine possiede una risoluzione molto elevata (1624x1234) per questo motivo ai fini di garantire una valutazione in tempo reale della data non è possibile effettuare un'analisi dell'intera immagine, ma saranno richiesti più passi di elaborazione che avranno il compito di avvicinarsi alla data fino a posizionarsi correttamente su di essa. Completata la fase di acquisizione è il sistema automatico che si occupa di effettuare il riconoscimento della data per verificare se la bombola deve essere nuovamente collaudata o meno. Nel caso in cui debba essere effettuato un nuovo collaudo la bombola viene separata da quelle che hanno superato positivamente il test. Le immagini registrate dal sensore non presentano tutte la stessa qualità, per questo motivo prima di iniziare la computazione è necessario adottare una fase di pre-elaborazione, che ha lo scopo di migliorare la qualità e di ridurre il più possibile i difetti contenuti. Questa operazione risulta essere molto importante, in quanto migliore sarà la pre-elaborazione e migliori saranno i risultati che si possono ottenere durante il riconoscimento della data. Completata questa fase iniziale, si procede all'individuazione dell'etichetta

utilizzando una tecnica di riconoscimento che sfrutta l'intensità luminosa dei pixel, in seguito una volta riconosciuta l'etichetta, si procederà all'individuazione del pittogramma (vedi Figura 1.2) .

Figura 1.2 Pittogramma

Tale operazione viene effettuata in quanto, come si può notare anche dalla Figura 1.1, il pittogramma è diametralmente opposto alla data, di conseguenza una volta individuata la sua posizione sarà semplicemente sufficiente spostarsi per localizzare la data. Completata l'individuazione si applicano tecniche invarianti a traslazione, cambiamenti di scala e rotazione per effettuare la classificazione dei numeri che costituiscono la data. Il progetto da me sviluppato, seppur essendo lontano da una possibile applicazione reale, vuole essere il più vicino possibile ad essa, per questo motivo tutte le tecniche utilizzate sono state sviluppate in modo che fornissero risultati *real time*.

1.2 Introduzione

Il riconoscimento automatico di oggetti (*patterns*) e la loro classificazione o raggruppamento (*clustering*) sono argomenti trattati in una grande varietà di problemi sia nell'area ingegneristica che in quella scientifica, come: la biologia, la psicologia, la medicina o la visione artificiale. Mentre per l'uomo risulta essere facile distinguere lettere o numeri scritti in corsivo o a macchina, diverso è il caso in cui questa attività di riconoscimento, detta appunto *Pattern recognition*,

debba essere automatizzata, ovvero effettuata in modo automatico da un computer [12]. Dato un *pattern*, che può essere per esempio un'impronta digitale, l'immagine di un volto o una parola scritta in corsivo [5], il suo riconoscimento, detto anche classificazione, consiste nell'individuare a quale classe appartiene il *pattern* stesso. Con classe si intende l'insieme delle entità che hanno proprietà comuni (come ad esempio i diversi modi in cui le persone scrivono la lettera "A"). A seconda dell'applicazione su cui si sta lavorando cambia il concetto di classe: si hanno 10 classi per il riconoscimento dei numeri oppure 2 classi per distinguere le vocali dalle consonanti [17]. La classificazione di un *pattern* può essere effettuata in due modi [12]:

- Classificazione supervisionata;
- Classificazione non supervisionata;

Nel primo caso il *pattern* fornito in input viene identificato come membro di una classe tra quelle predefinite, ovvero viene classificato in maniera supervisionata avendo l'utente o il progettista definito a priori le classe di interesse. In questo tipo di sistemi si utilizza infatti un *training set* (insieme di addestramento) dei dati. Tale termine viene usato per indicare un insieme di dati di cui si serve il sistema per effettuare l'apprendimento. Il *training set* consiste spesso di un vettore in input a cui viene associata una risposta o una determinata classificazione [16]. L'algoritmo apprende, in base alla risposta o alla classificazione fornita, le caratteristiche che discriminano gli elementi appartenenti alle differenti classi. Una volta completata questa fase di apprendimento, la correttezza dell'algoritmo viene verificata ripetendone l'esecuzione su un *test set*, che come dice il termine consiste di un insieme di esempi utilizzati per valutare le prestazioni del sistema [16], [17]. Uno dei principali problemi della classificazione supervisionata è costituito dal fenomeno dell'*overfitting* (sovradattamento). Come detto l'apprendimento viene effettuato fornendo in input all'algoritmo un *training set* di cui è già noto il risultato che ci interessa prevedere. Si assume che l'algoritmo di apprendimento raggiungerà uno stato in cui sarà in grado di predire la classificazione per tutti gli esempi che

ancora non ha visionato, cioè sarà in grado di effettuare una generalizzazione. Tuttavia, soprattutto nei casi in cui l'apprendimento è stato effettuato troppo a lungo o dove c'era uno scarso numero di *training set*, il modello potrebbe adattarsi a caratteristiche che sono specifiche dell'insieme di addestramento, ma che non hanno riscontro sul *test set*. Di conseguenza in presenza di *overfitting* le prestazioni sui campioni aumenteranno, mentre saranno peggiori quelle sui dati non visionati. Per risolvere questo problema esistono principalmente due tecniche: la *cross-validation* e l'arresto anticipato che indicano quando un ulteriore allenamento non porterebbe ad una migliore generalizzazione. Le due classi principali di algoritmi di apprendimento supervisionato sono le seguenti:

- Metodi Generativi;
- Metodi Discriminativi;

I metodi generativi si basano sulla creazione di un modello dei dati che poi viene utilizzato per predire le risposte desiderate. I metodi discriminativi, invece, cercano di modellare direttamente la relazione tra dati in input e dati in output, in modo da minimizzare una funzione di perdita, detta *loss function*.

Nella classificazione non supervisionata, a differenza di quella supervisionata, non si fa utilizzo del *training set*, in quanto il *pattern* in input viene assegnato ad una classe sconosciuta a priori e spetta al sistema stesso il compito di determinarla autonomamente a partire dai dati [17]. Le tecniche più utilizzate per questo tipo di classificazione fanno parte della famiglia del *Clustering* (raggruppamento). Con *clustering* si indica l'insieme delle tecniche che effettuano un raggruppamento di elementi che presentano similarità tra loro, a partire da un insieme di dati. In molti approcci questa similarità è ottenuta in termini di distanza in uno spazio multidimensionale. La qualità delle analisi effettuate da questo tipo di algoritmi dipende dalla scelta del modo in cui viene calcolata la distanza. Le tecniche di *clustering* raggruppano, quindi, gli elementi sulla base della loro distanza reciproca, di conseguenza l'appartenenza o meno ad un *cluster* (insieme) dipende da quanto l'elemento preso in considerazione è distante dall'insieme stesso. Ci sono differenti classificazioni dei tipi di

clustering ed esiste inoltre una certa dicotomia. La maggior parte delle soluzioni proposte possono essere ricondotte alle seguenti famiglie:

- Clustering Partitivo;
- Clustering Gerarchico;

Nel primo caso per definire l'appartenenza di un elemento ad un determinato gruppo si sfrutta la distanza da un punto rappresentativo del *cluster* (Es. centroide), avendo definito a priori il numero di gruppi della partizione risultato. Con il *clustering* gerarchico viene costruita una gerarchia di gruppi, visualizzabile tramite una rappresentazione ad albero detta dendrogramma, in cui vengono riportati i passi di accoppiamento/suddivisione dei gruppi.

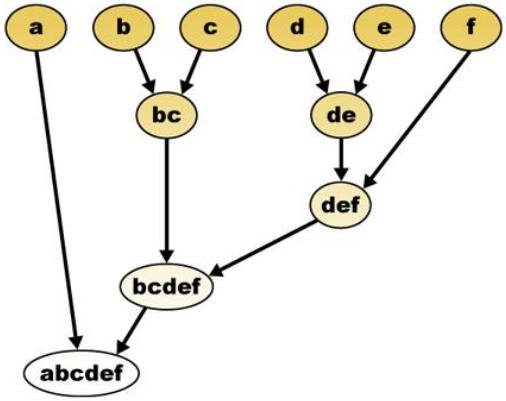


Figura 1.3 Esempio di dendrogramma

Le tecniche di *clustering* gerarchico possono essere ulteriormente classificate in:

- Metodi Aggregativi;
- Metodi Divisivi;

I metodi aggregativi seguono una procedura di tipo bottom-up dove inizialmente tutti gli elementi vengono considerati come appartenenti ad un unico *cluster*, in seguito l'algoritmo effettua l'unione dei *cluster* più vicini tra loro. Questa operazione viene ripetuta in modo iterativo fino ad ottenere un numero prefissato di cluster, o fino a che la distanza minima tra i *cluster* non supera una certa soglia. Con i metodi divisivi si procede in maniera esattamente contraria. Queste tecniche seguono una filosofia di tipo top-down in cui tutti gli elementi si trovano inizialmente in un unico *cluster*, l'algoritmo inizia poi a suddividere il *cluster* in

tanti *cluster* di dimensione inferiore. Questa operazione viene ripetuta fino a che non si raggiunge un numero prefissato di gruppi.

1.3 Classificazione dei sistemi di PR

Un generico sistema di *Pattern Recognition* può essere rappresentato nel seguente modo:

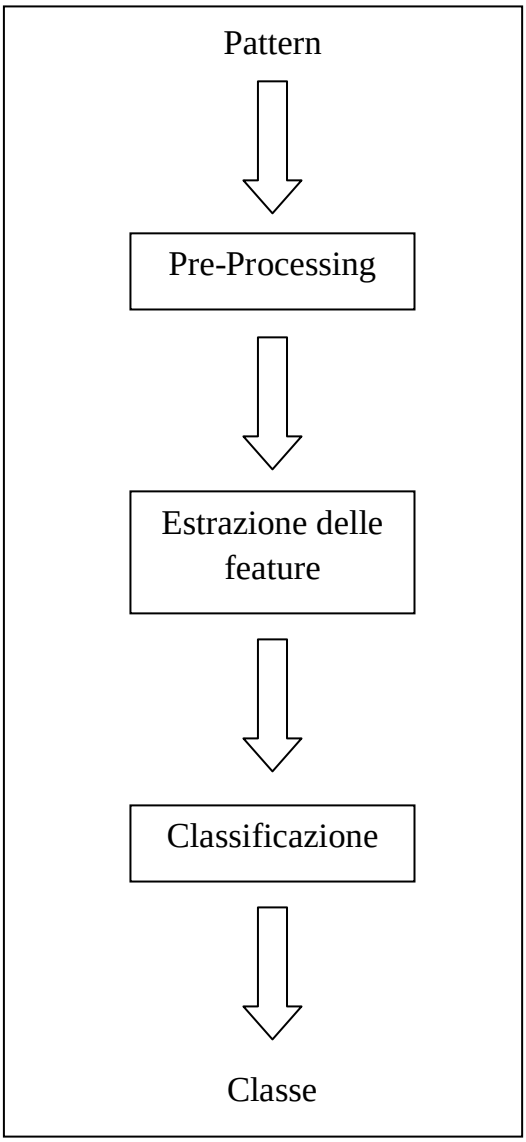


Figura 1.4 Schema generale di un sistema di Pattern recognition

Come si può notare dalla Figura 1.4 la progettazione di un generico sistema di *Pattern Recognition* richiede di affrontare i seguenti aspetti [12]:

1- Acquisizione e pre-elaborazione dei dati;

2- Estrazione delle *features* dei pattern forniti in input;

3- Decisione e classificazione dei dati;

I sistemi di questo tipo solitamente sono dotati di più telecamere utilizzate per effettuare l'acquisizione. Tale processo consiste nell'effettuare una digitalizzazione dell'immagine, realizzata tramite una conversione analogico-digitale effettuata dal dispositivo stesso. Attraverso questa fase si ottiene una rappresentazione digitale dell'immagine, chiamata rappresentazione *raster*. Tramite un campionamento spaziale si ottiene una matrice in cui ogni elemento rappresenta un pixel. La distanza (orizzontale e/o verticale) tra due pixel definisce la risoluzione alla quale l'acquisizione è stata effettuata. Una volta completata la fase di acquisizione si rende necessaria una fase di pre-elaborazione dell'immagine. Questa fase viene utilizzata per ridurre l'eventuale rumore prodotto in fase di acquisizione dal sensore e viene realizzata utilizzando tecniche di elaborazione dell'immagine. Per una maggiore trattazione di tali tecniche si rimanda ai capitoli successivi, in cui è presente una descrizione approfondita dei metodi di elaborazione principalmente utilizzati. La fase di estrazione delle *features* è una fase molto importante ai fini della classificazione. Visto che i dati in input risultano essere troppo grandi, o presentano eccessive ridondanze per essere elaborate dall'algoritmo, vengono trasformati in una ridotta rappresentazione di *features*. Questi dati prendono anche il nome di *features vector* in quanto possono essere rappresentati sotto forma di vettore. La procedura consiste nell'estrarre i dati più rilevanti a partire da quelli forniti in input. Alcune delle principali finalità di tale fase sono l'estrazione di dati invarianti per determinate trasformazioni (rotazione, scala, traslazione) o la loro normalizzazione per renderli maggiormente confrontabili. Obiettivo principale dell'estrazione delle caratteristiche è quindi semplificare la quantità di risorse necessarie per descrivere un grande insieme di dati con precisione. La loro

analisi, infatti, richiederebbe l'utilizzo di una eccessiva quantità di memoria e potenza di calcolo non presente nei calcolatori odierni. L'ultima fase è quella della classificazione, che come già descritto in precedenza ha il compito di determinare, a partire dalle *features* estratte per ogni *pattern*, la classe a cui appartiene. Una volta completata la descrizione di un generico sistema di *pattern recognition* ne effettuiamo una classificazione. Esistono quattro principali categorie [3]:

- Sistemi basati su modello (*Template Matching*);
- Sistemi con classificazione statistica (*Statistical Classification*);
- Sistemi sintattici o strutturali (*Syntatic or Structural Matching*);
- Sistemi basati su reti neurali (*Neural Networks*);

Approccio	Rappresentazione	Riconoscimento	Criterio di Classificazione
Basato su Modello (*Template Matching*)	Dati campionati, pixel, curve introduzione di distanze	Correlazione statistica,	Basato su stima dell'Errore di Classificazione
Classificazione Statistica (*Statistical Classification*)	Misure o Caratteristiche (Feature)	Funzioni discriminanti	Basato su stima dell'Errore di Classificazione
Sintattico o Strutturale (*Syntactic or Structural Recognition*)	Primitive	Regole, Grammatiche	Basato su stima dell'Errore di Accettazione
Reti Neurali (*Neural Network*)	Dati campionati, pixel, Caratteristiche (Feature)	Funzioni sinaptiche e di rete	Basato su stima dell'Errore quadratico medio

Figura 1.5 Classificazione dei sistemi di pattern recognition

Queste categorie non sono necessariamente separate ed indipendenti, infatti lo stesso metodo potrebbe essere utilizzato con approcci differenti. Esiste anche la possibilità di progettare sistemi ibridi, che sono il risultato dell'integrazione tra modelli differenti all'interno dello stesso sistema. Nei paragrafi successivi verranno descritte le caratteristiche delle quattro principali categorie dei sistemi di *pattern recognition*.

1.3.1 Sistemi basati su modello

Uno degli approcci più semplici è quello costituito dai sistemi basati su modello, detto anche *Template Matching*. Questo tipo di sistemi utilizzano la tecnica del *matching* che consiste nel determinare la similarità tra due entità dello stesso tipo. Il *template* è solitamente un prototipo, bidimensionale, dell'oggetto da riconoscere e classificare. La tecnica del *Template Matching* può essere ulteriormente suddivisa in altre due sottocategorie [1]:

- Corrispondenza basata sulle *features*;
- Corrispondenza basata sul *template*;

L'approccio basato sulle *features* sfrutta le caratteristiche dell'immagine modello, ad esempio bordi o orientazione del gradiente, come tecnica di *matching* per trovare la posizione migliore del *template* nell'immagine sorgente. Questo tipo di tecnica viene solitamente utilizzata quando l'immagine modello presenta delle caratteristiche ben evidenti. L'approccio basato su *template* utilizza l'intero modello su cui vengono effettuati calcoli come l'errore quadratico medio o la correlazione incrociata che determinano la migliore posizione del *template* verificando tutte o solo una parte delle posizioni fattibili all'interno dell'immagine sorgente. Questa tecnica è utilizzata quando il modello non presenta caratteristiche ben evidenti oppure quando esso costituisce la maggior parte dell'immagine di *matching*. L'approccio basato sulle *features* a differenza di quello basato su modello non utilizza l'intero *template* ma solo una piccola parte, ed è proprio per questo motivo che queste tecniche risultano essere computazionalmente più efficienti, in quanto le tecniche del secondo tipo potrebbero effettuare calcoli su un numero elevato di punti. Una possibile soluzione per migliore le prestazioni della corrispondenza basata su modello consiste nel ridurre la risoluzione dell'immagine sorgente e del *template* di uno stesso fattore e nell'effettuare le operazioni sulle nuove immagini ottenute, in modo tale da ridurre il numero di punti da esaminare.

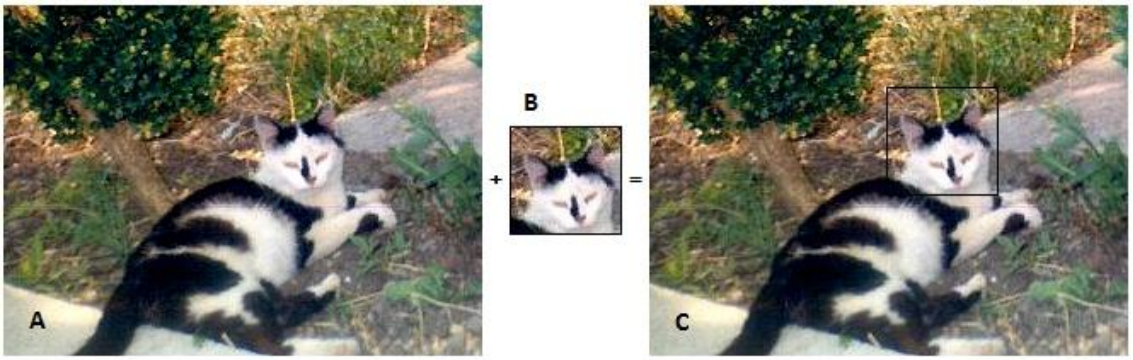


Figura 1.6 Esempio di sistema di pattern recognition basato su modello: A immagine sorgente, B immagine template, C immagine di output in cui si evidenzia la migliore posizione calcolata per il template

1.3.2 Sistemi con classificazione statistica

In questo tipo di sistemi il pattern è costituito da un vettore la cui dimensione costituisce il numero di *features* e viene rappresentato come un punto nello spazio multidimensionale delle caratteristiche. L'obiettivo di questi sistemi è quindi quello di selezionare un insieme di *features* che consentono a vettori appartenenti a classi differenti di occupare spazi compatti all'interno dello spazio delle caratteristiche.

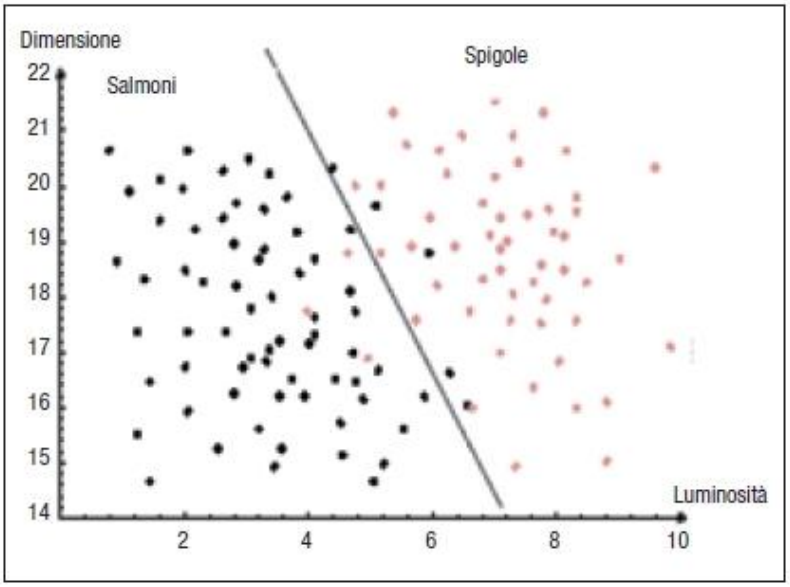


Figura 1.7 Spazio bidimensionale delle caratteristiche per due classi di pesci: i salmoni e le spigole

Uno degli esempi più utilizzati in letteratura consiste nell'effettuare la classificazione statistica di due categorie di pesci: i salmoni e le spigole [1]. Nella Figura 1.7 viene riportato uno spazio con due caratteristiche, la dimensione e la luminosità, utilizzate per distinguere immagini di pesci. Come si può notare i valori per ogni pesce si raggruppano in due regioni principali. Ognuno dei punti rappresentati nel grafico costituisce il vettore delle *features*, in questo caso bidimensionale perché si ha a che fare con due caratteristiche, del *pattern* fornito in input. L'efficacia della rappresentazione spaziale delle caratteristiche è rappresentata da come gli oggetti delle diverse classi sono separati nello spazio. Dato un *training set* per ogni classe, l'obiettivo è quello di costruire delle regioni di decisione nello spazio delle *features* in modo tale che i *pattern* appartenenti a classi diverse vengano correttamente separati e con il minor margine di errore. Il calcolo di queste regioni può essere effettuato in modo supervisionato (quindi specificate dall'utente) o non supervisionato (tramite addestramento del sistema); esse dipendono dalle distribuzioni di probabilità per ogni oggetto di appartenere ad una delle classi. Una delle tecniche più utilizzate per il calcolo consiste nello scegliere, in modo intuitivo, la migliore forma parametrica per la regione di decisione, in seguito si sceglie, sulla base dei dati del *training set*, la migliore regione per effettuare la classificazione. Solitamente si utilizzano dei metodi di natura statistica per la stima degli errori per determinare la regione di decisione ed il motivo è dato dal fatto che viene scelta come migliore regione quella che minimizza l'errore di assegnazione di un *pattern* ad una classe sbagliata.

1.3.3 Sistemi con classificazione sintattica o strutturale

Quando si affrontano problemi di *pattern recognition* che coinvolgono oggetti complessi, una possibile soluzione consiste nell'adottare una suddivisione gerarchica, in cui gli oggetti vengono considerati come composti da sottoparti più semplici [2], [4]. I metodi di riconoscimento di questa categoria consentono quindi di semplificare la struttura del *pattern*. Un oggetto complesso è formato da un insieme di *subpatterns* più semplici che possono essere a loro volta

ulteriormente suddivisi. Le parti più piccole che costituiscono l'oggetto prendono il nome di primitive, mentre i legami che esistono tra esse sono detti relazioni. L'approccio strutturale consente quindi di descrivere un ampio insieme di *patterns* complessi, utilizzando parti più piccole e regole di composizione per esse. Questo tipo di classificazione coinvolge tre processi indipendenti:

- Estrazione e identificazione delle primitive;
- Identificazione delle relazioni esistenti tra le primitive;
- Identificazione di strutture valide in termini di primitive e di relazioni tra esse;

In questi sistemi esiste un'analogia tra la struttura dei *patterns* complessi e la sintassi di un linguaggio. I *patterns* sono concepiti come frasi di un linguaggio, mentre i *subpatterns* come l'alfabeto del linguaggio, di conseguenza le frasi sono generate utilizzando la grammatica del linguaggio. L'oggetto viene quindi classificato solo se la corrispondente frase appartiene al linguaggio specificato per la classe ed è perciò accettata dalla grammatica specificata. La grammatica per ogni classe di *patterns* può essere inferita o appresa dall'insieme di addestramento.

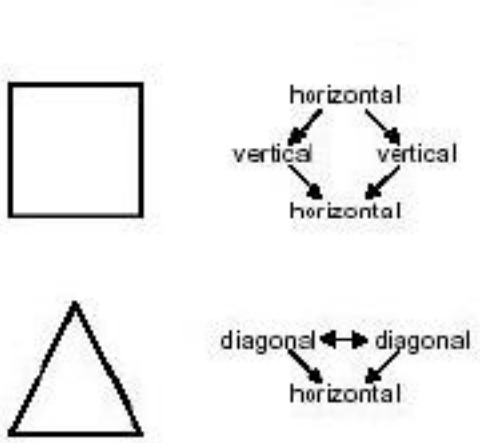


Figura 1.8 Esempio di approccio statistico

Come si può vedere dalla Figura 1.8 l'approccio strutturale fornisce oltre alla classificazione dell'oggetto anche una descrizione di come esso sia costruito a partire dalle primitive. Nell'esempio sopra riportato vengono estratte le caratteristiche morfologiche e le loro relazioni.

1.3.4 Sistemi basati su reti neurali

Il cervello umano è costituito da una rete biologica formata da milioni di cellule collegate tra loro, dette neuroni. Questi neuroni attraverso le interconnessioni si scambiano informazioni ed il cervello è in grado di apprendere analizzare e riconoscere. Le reti neurali, sono tecniche computazionali fortemente parallele che cercano di simulare all'interno di un sistema informatico, il comportamento del cervello umano. Anche esse sono costituite da un gran numero di cellule o neuroni collegati fra loro a formare una rete complessa.

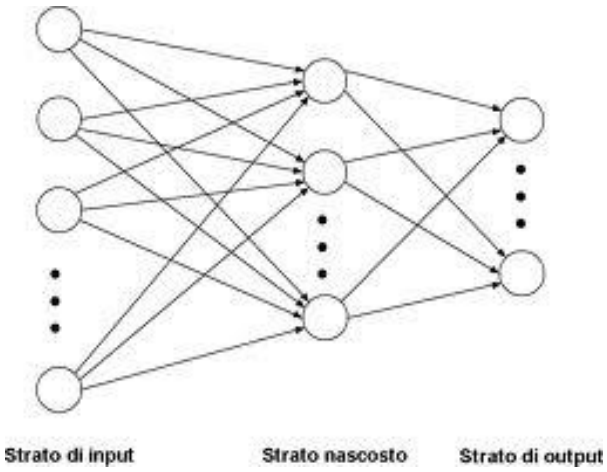


Figura 1.9 Rete neurale

I modelli basati su reti neurali possono essere rappresentati tramite grafi: i nodi costituiscono i neuroni artificiali, mentre gli archi rappresentano le connessioni tra l'output di un livello di neuroni e l'input del livello successivo (vedi Figura 1.9). Solitamente le reti sono formate da tre strati: il primo strato è quello degli *input*, che si occupa di trattare i dati in ingresso in modo da renderli compatibili alle richieste dei neuroni. Il secondo strato è quello nascosto (*hidden*) che effettua l'elaborazione vera e propria e può essere formato anche da più livelli di neuroni, mentre l'ultimo strato è quello dell'*output* che si occupa di raccogliere i risultati provenienti dai nodi interni e di adattarli per le successive fasi della computazione. Le singole unità che compongono la rete neurale sono fondamentali ai fini del corretto funzionamento del sistema. Una possibile loro rappresentazione è la seguente:

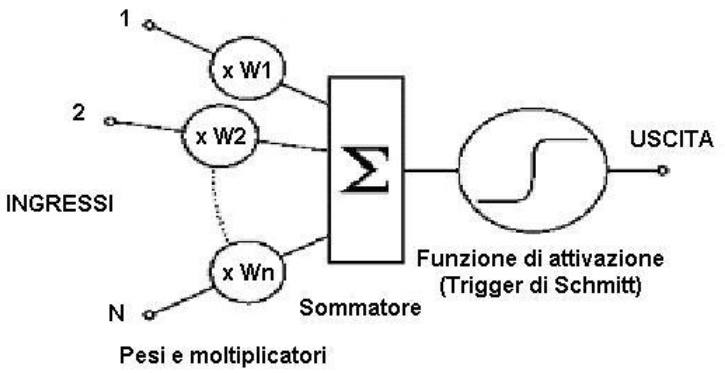


Figura 1.10 Modello di neurone artificiale

Ogni neurone possiede un certo numero di ingressi, attraverso cui ricevono informazioni dal sistema, ma una sola uscita, grazie alla quale emettono risposte nella rete. Quando l'unità riceve in *input* un segnale inizia la computazione. In un caso semplicistico si può pensare che per ognuno degli ingressi viene effettuato il prodotto per un opportuno valore detto peso, in seguito viene effettuata una somma pesata di tutte le entrate e se il risultato supera una certa soglia il neurone attiva la sua uscita. Il peso viene utilizzato per quantificare l'importanza di un ingresso, di conseguenza un ingresso di elevata importanza avrà un peso elevato, viceversa avrà un peso inferiore. Questi sistemi di *pattern recognition* stanno assumendo sempre maggiore importanza, ed il motivo è dato dal fatto che le reti neurali sono indipendenti dal dominio applicativo del sistema ed inoltre esistono algoritmi efficienti per l'apprendimento da *training set* dei *patterns* da classificare.

1.4 Optical Character Recognition

Il riconoscimento ottico dei caratteri è un tema di ricerca molto studiato nel campo del *pattern recognition* [11]. Questi sistemi hanno lo scopo di consentire al calcolatore il riconoscimento automatico dei caratteri, presenti in forma digitale, nelle immagini. Inizialmente la lettura ottica (usando tecniche ottiche

come gli obiettivi) e il riconoscimento digitale dei caratteri (usando algoritmi di separazione ed analisi del testo) venivano considerati come due campi separati, in quanto vi erano numerose distinzioni, tuttavia la quasi scomparsa delle applicazioni del primo tipo, ha portato all'utilizzo del termine OCR per indicare il riconoscimento digitale delle immagini indipendentemente dalla sorgente utilizzata. Queste applicazioni, con il passare degli anni, hanno trovato sempre un maggior numero di impieghi, come la guida automatica di veicoli, le librerie digitali, o recentemente nei palmari. Sono stati proposti numerosi algoritmi per ottenere migliori risultati durante il riconoscimento, alcuni di questi sono il *template matching*, le *features* geometriche o gli invarianti di immagine basati su forma. Tra tutti gli algoritmi proposti, quest'ultimi risultano essere i più interessanti, in quanto presentano la proprietà di invarianza, che possono migliorare le prestazioni del riconoscimento anche nel caso in cui l'immagine ha subito una particolare trasformazione, come scalatura, traslazione o rotazione. Queste tecniche si suddividono in due categorie:

- *Boundary-based* (basate su contorno);
- *Region-based* (basate su regione);

La prima categoria di invarianti si concentra sulle proprietà contenute nei contorni dell'immagine, mentre la seconda categoria prende l'intera area dell'immagine come obiettivo della ricerca. Lo svantaggio di queste ultime tecniche è dato proprio dal fatto che esse prendono in considerazione l'intera immagine, di conseguenza il numero totale di pixel contenuti nell'immagine risulta essere un fattore decisivo per il costo computazione dell'algoritmo. Un altro problema che in questo caso riguarda tutte le tecniche, è dato dal rumore che viene introdotto durante il processo di acquisizione. L'immagine che viene catturata rappresenta, infatti, solo una versione degradata di quella originale. A seconda delle tecniche che vengono utilizzate si ottengono risultati più o meno buoni, che dipendono da quanto l'algoritmo è influenzato dal rumore presente all'interno dell'immagine. La struttura di un sistema di OCR è la seguente:

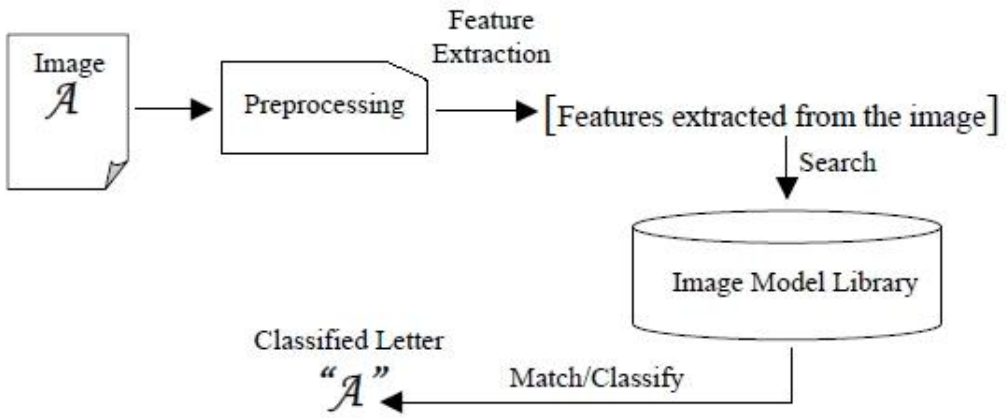


Figura 1.11 I processi base di un sistema OCR

Come si può notare dalla Figura 1.11 lo schema di un OCR risulta essere lo stesso del modello generale mostrato in Figura 1.4. In questo caso viene anche riportato l'*Image Model Library*, che viene utilizzato durante la ricerca della corrispondenza tra le *features* dei caratteri estratti dall'immagine originale ed i modelli (come ad esempio *template*) contenuti in esso. In questo modo il sistema è in grado di estrarre autonomamente il corretto modello, a seconda delle *features* che rappresentano il carattere in esame. I sistemi di OCR possono essere ulteriormente classificati in due gruppi:

- *Handwritten character recognition*;
- *Printed character recognition*;

Le tecniche appartenenti alla prima categoria effettuano il riconoscimento di caratteri scritti a mano, mentre quelli della seconda categoria identificano caratteri stampati. Il riconoscimento di caratteri scritti a mano rispetto a quelli stampati risulta essere un problema molto più complesso, in quanto ogni persona possiede un proprio stile, ed è necessario analizzare e studiare le diverse possibilità con cui ogni umano potrebbe scrivere un carattere.

Capitolo 2

Le fasi di un sistema di riconoscimento

In questo capitolo vengono analizzate e descritte le diverse fasi che compongono un sistema di *pattern recognition* a partire dal pre-processing, estrazione delle *features* e classificazione.

2.1 Acquisizione e rappresentazione di immagini digitali

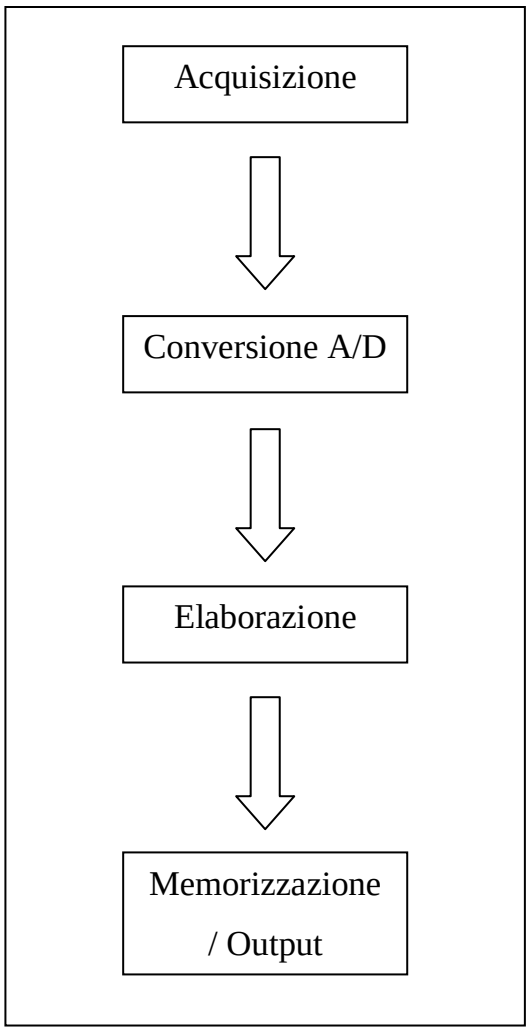


Figura 2.1 Schema di un generico sistema di elaborazione dell'immagine

L’acquisizione di immagini digitali costituisce il primo passo di un sistema di elaborazione dell’immagine [6]. Questa operazione consiste nell’effettuare una conversione analogico/digitale dell’immagine. Ci sono due principali modalità di acquisizione:

- Modalità analogica;
- Modalità digitale;

Nella modalità analogica, utilizzata ad esempio nelle vecchie fotocamere analogiche, il processo di digitalizzazione avviene successivamente a quello dell’acquisizione, in quanto il dispositivo non è adottato di un convertitore A/D, mentre nella modalità digitale le due operazioni vengono considerate come uniche, perché la fase di conversione viene effettuata dal dispositivo stesso. Il grande vantaggio dei sistemi digitali a discapito di quelli analogici è proprio dato dal fatto che la velocità computazionale e le prestazioni dei processori integrati hanno raggiunto un livello per cui l’operazione di acquisizione e conversione può essere effettuata entro pochi secondi. L’ampia diffusione negli ultimi anni di dispositivi digitali, ha portato alla quasi scomparsa dei sistemi analogici. Completata la fase di digitalizzazione si applicano opportuni algoritmi per migliorare la qualità dell’immagine finale, infine si effettua la memorizzazione o la stampa su un dispositivo di output. In caso di memorizzazione i dati possono essere codificati per ridurre lo spazio occupato dall’immagine. Le informazioni riguardanti le impostazioni di acquisizione possono essere incluse all’interno dei dati per favorire una corretta interpretazione dell’immagine. Quando l’immagine viene riportata in output, queste informazioni vengono combinate con le informazioni del dispositivo di output per produrre l’immagine migliore per gli scopi dell’utente.

2.1.1 Sensori

L’elemento fondamentale che consente di effettuare l’acquisizione di una immagine è il sensore. Questo componente, integrato all’interno del dispositivo di acquisizione, può essere visto come una matrice di piccoli elementi sensibili

alla luce detti foto-elementi che producono un segnale elettrico proporzionale al livello di energia registrato. L'insieme del valori forniti dai foto-elementi viene elaborato dal sensore e costituisce l'informazione analogica che rappresenta l'immagine. Esistono due principali tecnologie per la realizzazione di sensori [6]:

- Sensori CCD;
- Sensori CMOS;

I sensori CCD (acronimo di *Charge Coupled Device*) sono costituiti da una griglia di foto-elementi accoppiati tra di loro, in modo tale che ognuno di essi, sollecitato da un impulso elettrico, possa trasferire la propria carica ad un elemento adiacente [6].

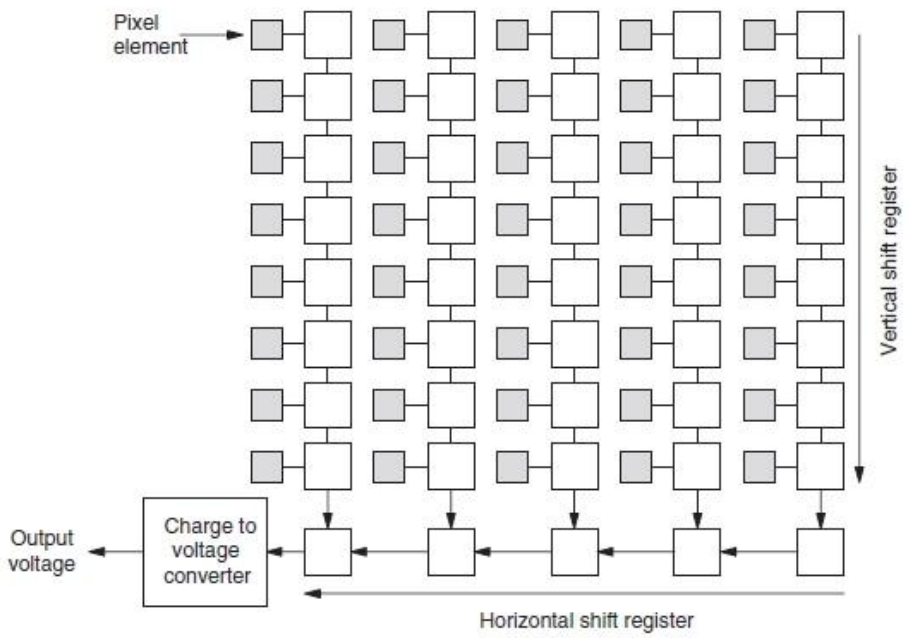


Figura 2.2 Diagramma di un sensore CCD

Come si può vedere dalla figura, la carica di ogni singolo foto-elemento viene prima memorizzata in registri a scorrimento verticale, poi viene trasferita su registri a scorrimento orizzontale ed infine inviata verso l'uscita senza che venga effettuato alcun tipo di trattamento [6]. Il segnale in output dal sensore è un segnale analogico, di conseguenza è presente un convertitore che si occupa della digitalizzazione. Nei sensori CMOS (acronimo di *complementary metal-oxide*

semiconductor), a differenza dei CCD, ogni foto-elemento è dotato di un convertitore carica elettrica-voltaggio e sono inoltre presenti amplificatori, circuiti di riduzione del rumore e di digitalizzazione in modo tale che il segnale in uscita sia già stato convertito in digitale (vedi Figura 2.3) [6], [15].

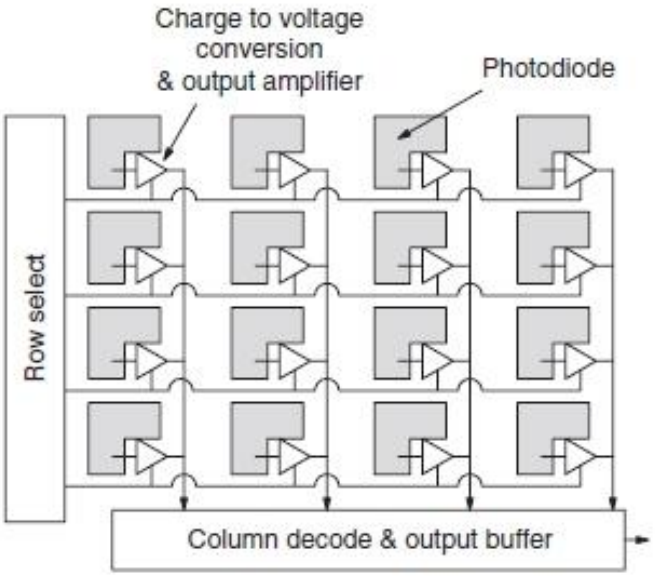


Figura 2.3 Diagramma di un sensore CMOS

Questo tipo di sensore ha, rispetto al CCD, una parte più ampia della superficie dedicata ad elementi circuitali che non si occupano di catturare la luce, quindi l'area effettivamente utilizzabile per la rilevazione dell'informazione luminosa è inferiore [15]. Inoltre visto che la conversione viene effettuata da ogni singolo pixel si ha una uniformità del segnale più bassa rispetto a quella ottenuta con i sensori CCD. Ad oggi la tecnologia CCD tende ad essere utilizzata in applicazioni di fascia alta, che richiedono un rumore ridotto, come ad esempio fotocamere digitali di alta qualità, mentre i sensori CMOS sono utilizzati in grandi volumi e con costi di produzione ridotti, nelle fotocamere per cellulari.

2.1.2 Digitalizzazione

Una immagine può essere considerata come una funzione f(x,y) bi-dimensionale continua rappresentante una scena. Con (x,y) indichiamo le coordinate spaziali

dell'immagine, mentre con f(x,y) indichiamo la sua intensità luminosa nel punto (x,y) [15]. Per poter essere elaborata dal calcolatore si deve convertire l'immagine analogica in una rappresentazione numerica della scena attraverso un processo di digitalizzazione. Questo processo consiste nel discretizzare l'immagine sia nelle coordinate spaziali (x,y), sia nell'intensità luminosa f(x,y) ed è formato da due operazioni fondamentali:

- Campionamento;
- Quantizzazione;

Il campionamento costituisce il primo passo per la conversione di un segnale da analogico a digitale. L'immagine reale viene campionata, ovvero si individuano alcuni punti significativi di essa, che ci consentono di poterla ricostruire con una buona approssimazione. Questa operazione sfrutta il teorema di Nyquist-Shannon che stabilisce quali sono i criteri di scelta del numero e della posizione dei punti da utilizzare per campionare l'immagine. Il teorema del campionamento prevede l'utilizzo di una frequenza almeno doppia rispetto alla massima frequenza del segnale da campionare. Tale frequenza minima di campionamento prende il nome di frequenza di Nyquist. Nel caso in cui la condizione non venga rispettata, ovvero si effettua un campionamento ad una frequenza inferiore a quella minima, si verificherà il problema dell'alias, in cui le alte frequenze del segnale originale verranno mappate come basse frequenze del segnale campionato.

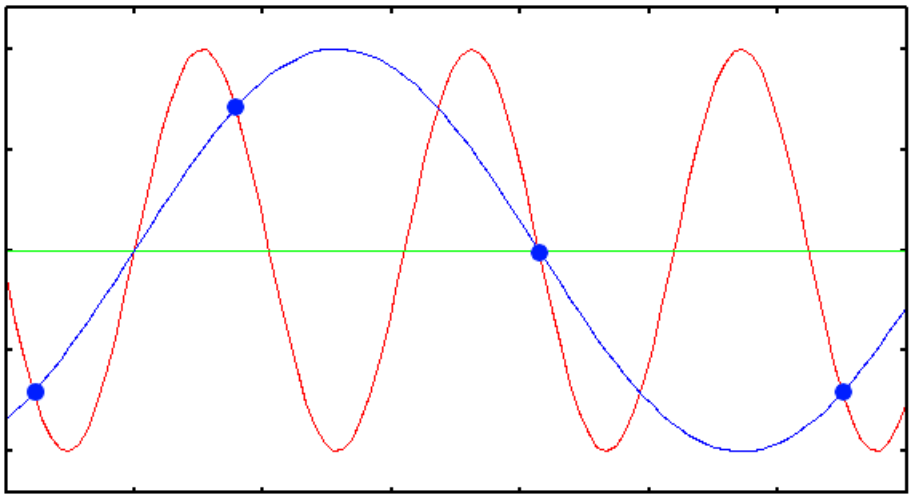


Figura 2.4 Problema dell'alias dovuto al sottocampionamento dell'immagine

Per risolvere questo problema si può utilizzare un filtro che elimina dal segnale originale tutte quelle frequenze al di sopra della metà della frequenza di Nyquist. Questo filtro, per la funzione che svolge, prende il nome di filtro anti-alias. Esso deve essere utilizzato prima della fase di campionamento, perché una volta che il segnale è stato sottocampionato e quindi dopo la comparsa dell'alias, non vi è più modo di eliminare l'effetto indesiderato.

Affinché l'immagine sia trasmissibile in forma numerica, è necessario che possa assumere solo un numero finito di valori (bit), questo viene effettuato tramite un processo di quantizzazione. I valori dell'immagine vengono limitati tra un massimo ed un minimo di valori discreti definiti preventivamente, in questo modo il dato analogico originale, in corrispondenza del campione dell'immagine, verrà ricondotto al più vicino dei valori discreti definiti in precedenza. Con questa operazione viene commesso un errore di quantizzazione, dovuto all'approssimazione del segnale analogico con un numero discreto di valori, tale problema può essere parzialmente ridotto all'aumentare del numero di valori utilizzati per effettuare la quantizzazione. La conversione del segnale da analogico a digitale viene poi completata codificando i dati discreti. In output al processo di digitalizzazione si ottiene, quindi, una immagine digitale che può essere vista come una matrice MxN di valori discreti. Ogni elemento della matrice prende il nome di pixel (*Pictures Elements*) [15].

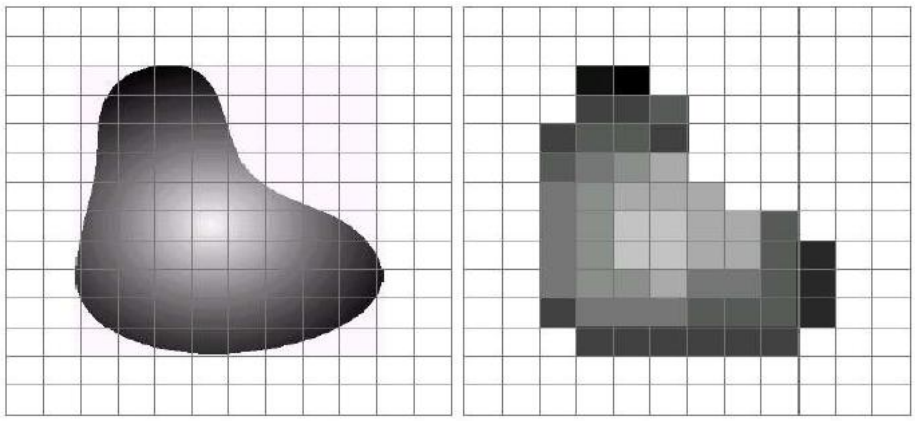


Figura 2.5 Immagine prima e dopo la digitalizzazione

Esistono due formati grafici principali per la rappresentazione di immagini digitali [15]:

- Grafica *raster* (o bitmap);
- Grafica vettoriale;

Con la prima tecnica l'immagine viene considerata come una matrice di punti (pixel). In ogni locazione della matrice viene memorizzata l'intensità luminosa del pixel. Questo tipo di immagini sono caratterizzate da due proprietà:

- Risoluzione;
- Profondità di colore;

La risoluzione indica il numero di pixel contenuti nell'unità di misura considerata (generalmente il pollice) e si ottiene moltiplicando il numero di righe della matrice per il numero delle colonne. La profondità stabilisce il numero di bit utilizzati da ogni pixel per descrivere il colore. Le immagini di tipo vettoriale sono descritte tramite primitive geometriche che definiscono punti, curve e poligoni ai quali si possono attribuire colori o sfumature. Questo tipo di immagini rispetto a quelle *raster* occupano un minor spazio in memoria e sono particolarmente adatte nel caso si debbano effettuare modifiche in quanto, a differenza della controparte bitmap, non si ha perdita di qualità su operazioni come l'ingrandimento dell'immagine. Le immagini *raster*, grazie anche alla loro semplicità di rappresentazione, rimangono quelle più utilizzate.

2.1.3 Tipi di immagine

Esistono tre principali tecniche per la memorizzazione di immagini bitmap:

- Immagini binarie;
- Immagini *grey-level*;
- Immagini a colori;

Le immagini binarie sono caratterizzate dal possedere solo due possibili livelli di grigio, rappresentati da 1 e 0 che indicano rispettivamente la luce (bianco) o l'assenza di luce (nero). Le immagini *grey-level* possiedono più livelli di grigio, il loro numero viene rappresentato tramite l'utilizzo di un intervallo. Solitamente

si utilizzano immagini a 256 livelli di grigio (da 0 a 255) anche se l'occhio umano non è in grado di distinguerne più di 64. L'ultima categoria riguarda le immagini a colori, in questo caso viene utilizzato come modello di riferimento il sistema RGB che si basa sui tre colori rosso, verde e blu. Questa categoria può essere ulteriormente suddivisa in altre due sottocategorie, le immagini *truecolor* e quelle indicizzate. Per le immagini *truecolor* si utilizzano tre matrici differenti, ognuna delle quali contiene i valori dell'immagine per uno dei tre colori del modello RGB, l'immagine risultante si ottiene combinando le matrici tra di loro. Per la seconda sottocategoria si utilizza una sola matrice i cui valori non sono altro che puntatori ad una tavolozza che memorizza l'elenco dei colori possibili.

2.1.4 Demosaicing

Il sensore del dispositivo di acquisizione ha il compito di raccogliere la luce tramite i foto-elementi e di convertirla in un segnale elettrico sfruttando la circuiteria interna, che lo elaborerà per poter estrarre l'immagine finale. Il problema è causato dal fatto che i foto-elementi non sono sensibili al colore, ma solo all'intensità luminosa, di conseguenza il sensore è semplicemente un dispositivo monocromatico [15]. Una possibile soluzione consisterebbe nell'utilizzare tre sensori differenti per ogni foto-elemento, ognuno dei quali si occupa della cattura di uno dei tre colori RGB. Questo posizionamento non risulta però semplice ed inoltre comporterebbe costi di produzione elevati, per questo motivo si utilizza un solo sensore per foto-elemento ottenendo una griglia di valori corrispondenti ai diversi colori RGB. Tale griglia prende il nome di *color filter array* (CFA). Questo filtro viene sovrapposto ai foto-elementi in modo da renderli sensibili alla luce di un particolare colore [15]. Esistono numerosi schemi per il CFA, ma il più diffuso e usato nelle fotocamere digitali è una combinazione RGB nota come *Bayer Pattern*. Esso misura l'immagine verde su una griglia a scacchiera, mentre quella rossa e quella blu su griglie rettangolari. Si effettua un maggior campionamento sul colore verde, in quanto l'occhio umano è più sensibile alle piccole variazione di lunghezza d'onda del

verde [15]. Esistono numerosi altri schemi oltre a quello appena citato, ma per brevità sono stati riassunti nella figura seguente

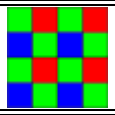
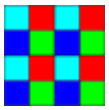
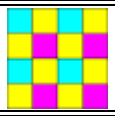
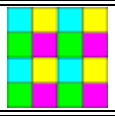
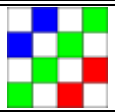
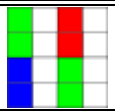
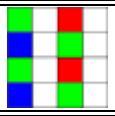
Immagine	Nome	Descrizione	Pattern (pixel)
	Filtro Bayer	Comune filtro RGB. La proporzione è uno rosso, uno blu e due verdi.	2×2
	Filtro RGBE	Simile al filtro di Bayer con uno dei filtri Verdi modificato in smeraldo. Usato in alcune fotocamere Sony.	2×2
	Filtro CYYM	Uno ciano, due gialli e uno magenta. Filtro usato in alcune fotocamere Kodak.	2×2
	Filtro CYGM	Uno ciano, uno giallo, uno verde e uno magenta.	2×2
	Filtro RGBW Bayer	Tradizionale filtro RGB con l'aggiunta del bianco, simile al filtro di Bayer e a quello RGBE.	2×2
	RGBW #1	Tre esempi di filtri RGBW da Kodak, con il 50% di bianco.	4×4
	RGBW #2		
	RGBW #3		2×4

Figura 2.6 Lista dei diversi schemi per il color filter array

L'utilizzo del *color filter array* non è pero sufficiente per la corretta ricostruzione di una immagine a colori. In questo modo infatti ogni foto-elemento viene reso sensibile alla luce di una particolare intensità, mentre come sappiamo, ogni pixel nelle immagini a colori è costituito da una tripla RGB. Per questo motivo è necessario un ulteriore processo chiamato *demosaicing*. Tale operazione è solitamente realizzata tramite l'utilizzo di algoritmi che consentono di ricostruire la rappresentazione a colori di una immagine a partire dai dati grezzi acquisiti da un sensore con CFA. Questi algoritmi sfruttano l'interpolazione per generare in ogni pixel le due componenti cromatiche mancanti. Le tecniche maggiormente

utilizzate, che sono anche quelle più efficienti, consistono nell'utilizzare metodi ad hoc, progettati in base alla struttura del CFA.

2.2 Tecniche di elaborazione delle immagini

Le tecniche che descriveremo successivamente fanno parte della fase di pre-processing di un sistema di *pattern recognition* (vedi Figura 1.4). Come sappiamo esse sono utilizzate per migliorare la qualità dell'immagine acquisita tramite sensore, o per evidenziare caratteristiche dell'immagine importanti per la successiva fase di estrazione delle caratteristiche. Gli approcci utilizzati possono essere divisi in due categorie [9], [15]:

- *Spatial domain approach* (Approccio nel dominio spaziale);
- *Frequency domain approach* (Approccio nel dominio delle frequenze);

Con dominio spaziale si intende l'insieme dei pixel che costituiscono un'immagine, quindi questo tipo di tecniche operano direttamente sui pixel dell'immagine. Le funzioni di elaborazione possono essere espresse come:

$$g(x,y) = T[f(x,y)]$$

dove $f(x,y)$ rappresenta l'immagine originale, $g(x,y)$ è l'immagine elaborata, mentre T è un operatore su f definito in un intorno di (x,y). Esistono due categorie che si differenziano a seconda della dimensione dell'intorno (x,y) [9], [15]:

- Tecniche di elaborazione puntuale (*Point processing*);
- Tecniche di elaborazione locale (*Area processing*);

Le tecniche appartenenti alla seconda categoria operano nel dominio delle frequenze, precisamente nel dominio di Fourier. Esse consistono nell'effettuare la trasformata di Fourier dell'immagine originale, in modo da passare dal dominio spaziale a quello delle frequenze, nell'applicare tecniche di filtraggio ed infine nell'utilizzare la trasformata di Fourier inversa per ritornare nel dominio spaziale. Nel seguito verranno descritte le principali tecniche che operano nel

dominio spaziale, in quanto sono quelle realmente utilizzate al fine della realizzazione del progetto di tesi.

2.2.1 Tecniche di elaborazione puntuale

In questo tipo di tecniche il valore di ogni pixel viene modificato solo in base al valore del pixel dell'immagine originale. Nessun altro pixel viene coinvolto nella modifica, di conseguenza per questa categoria l'intorno in cui viene definito l'operatore che effettua la trasformazione coincide con il pixel stesso. Se il risultato dell'elaborazione dipende solo dal valore del pixel a cui è applicata si ha a che fare con elaborazioni puntuali omogenee, altrimenti se il risultato dipende anche dalla posizione del pixel vengono chiamate elaborazioni puntuali non omogenee. Le tecniche principalmente utilizzate sono le seguenti [9], [15]:

- *Image brightening* (luminosità);
- *Image negative* (negativo);
- *Image thresholding* (binarizzazione);
- *Image contrast stretching* (normalizzazione);

Image brightening

Si aggiunge o sottrae un valore costante ad ogni pixel che costituisce l'immagine di partenza. Il risultato non è altro che un aumento o una diminuizione dell'intensità dell'immagine.

Figura 2.7 Immagine prima e dopo l'applicazione dell'image brightening

Image negative

Si sottrae ad ogni pixel dell'immagine il massimo valore consentito. Il risultato che si ottiene sarà che i valori vicini allo zero, verranno trasformati in pixel con intensità prossima al massimo e viceversa. Questa operazione può anche essere effettuata utilizzando un valore di soglia che indica a partire da quale valore i pixel devono essere negati.

Figura 2.8 Immagine prima e dopo l'applicazione dell'image negative

Image tresholding

Questa operazione trasforma una qualsiasi immagine a colori o a scala di grigio in bianco e nero. Tutti i pixel che non superano un certo valore di soglia vengono trasformati a nero, mentre quelli che la superano sono convertiti a bianco. Per questa tecnica risulta molto importante la scelta del valore di soglia per evitare di perdere eccessiva informazione durante la binarizzazione.

Figura 2.9 Immagine prima e dopo la binarizzazione

Image contrast stretching

Questa tecnica viene solitamente utilizzata per aumentare il range dinamico dei livelli di grigio nelle immagini a basso contrasto. Per esempio se abbiamo una immagine con un range di intensità che varia tra 50 e 150, mentre il range desiderato va da 0 a 200, il processo sottrae 50 dall'intensità di ogni pixel, facendo così variare l'intervallo da 0 a 100. A questo punto ogni pixel viene moltiplicato per 200/100, ottenendo così il range desiderato.

Figura 2.10 Immagine prima e dopo la normalizzazione

2.2.2 Tecniche di elaborazione locale

Queste tecniche producono modifiche nei livelli di grigio di ogni pixel dell'immagine originale, basandosi sul comportamento dei pixel ad esso adiacenti in un intorno più o meno grande. In questo caso l'intorno coincide con una matrice bi-dimensionale di pixel, chiamata maschera, di dimensione dispari e centrata sul pixel in esame [9]. Le tecniche di questa categoria sono realizzate sfruttando la maschera (detta anche kernel o filtro) ed applicando l'operazione della convoluzione discreta. In matematica la convoluzione è un'operazione (simbolo *) che, date due funzioni, ne restituisce una terza così definita:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

Nel campo della grafica f e g corrisponderanno ad immagini, saranno quindi bi-dimensionali e rappresentabili con funzioni a valori discreti su domini limitati, per questo motivo si parla di convoluzione discreta.

Data una immagine $f(x,y)$ di dimensioni NxN e sia $k(x,y)$ la maschera di dimensione nxn l'immagine ottenuta tramite convoluzione discreta è la seguente:

$$g(x,y) = k(x,y) * f(x,y) = \sum_{i=-a}^{a} \sum_{j=-a}^{a} k(i,j) f(x-i, y-j)$$

dove a=(n-1)/2, x,y=0,…,N-1 e l'intorno del punto (x,y) è della stessa dimensione del nucleo [9].

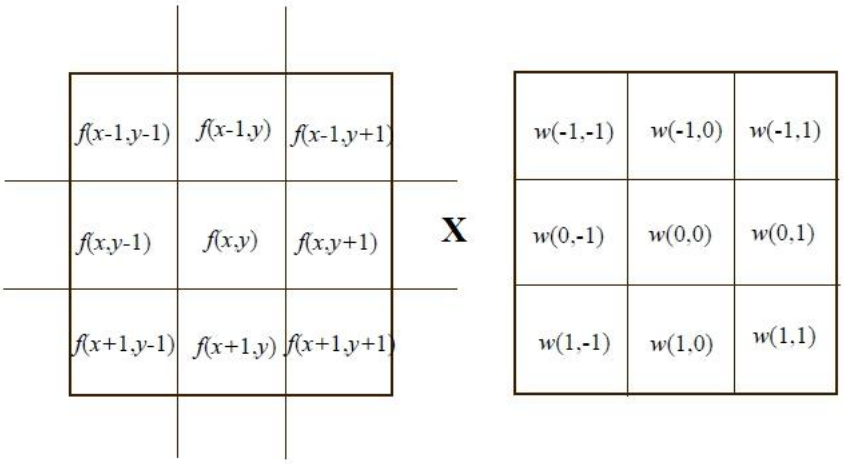


Figura 2.11 Nucleo di convoluzione: a sinistra è presente la matrice estratta dall'immagine originale, mentre a destra viene riportata la matrice rappresentante il nucleo di convoluzione

In altri termini questa operazione equivale a moltiplicare i valori dei pixel dell'immagine da filtrare per i corrispondenti coefficienti della maschera di convoluzione (vedi Figura 2.11). In seguito i risultati vengono poi sommati insieme a formare il risultato del pixel centrale nella matrice estratta dall'immagine originale (nella Figura 2.11 il pixel $f(x,y)$). L'operazione viene poi ripetuta, spostando la matrice, per tutti i pixel dell'immagine. Esistono due principali categorie di tecniche di elaborazione locale [9], [15]:

- Tecniche di regolarizzazione (*Image smoothing*);
- Tecniche per evidenziare i bordi (*Image sharpening*);

Tecniche di regolarizzazione

Sono metodi che tengono conto del peso di una zona relativamente ampia nell'intorno del pixel in esame. Hanno l'obiettivo di rendere l'immagine più

regolare, eliminando il rumore dovuto al processo di acquisizione. L'idea è quella di cercare il colore che presenta peso più elevato nell'intorno del pixel esaminato che verrà poi sostituito con quest'ultimo. Il modo in cui vengono scelti i pesi differenzia i vari algoritmi. I più importanti sono:

- Tecnica della media;
- Valore mediano;

Con la tecnica della media il valore del nuovo pixel viene calcolato come media del suo intorno. Si sovrappone la maschera all'immagine e si calcola la media dei pixel facenti parte del nucleo. Il risultato ottenuto viene poi sostituito al valore del pixel centrale. La tecnica del valore mediano, funziona nello stesso modo della prima, in questo caso però anziché sostituire il valor medio si controllano tutti punti della maschera di convoluzione, li si ordinano ed infine si sceglie il valore che si trova nella posizione mediana.

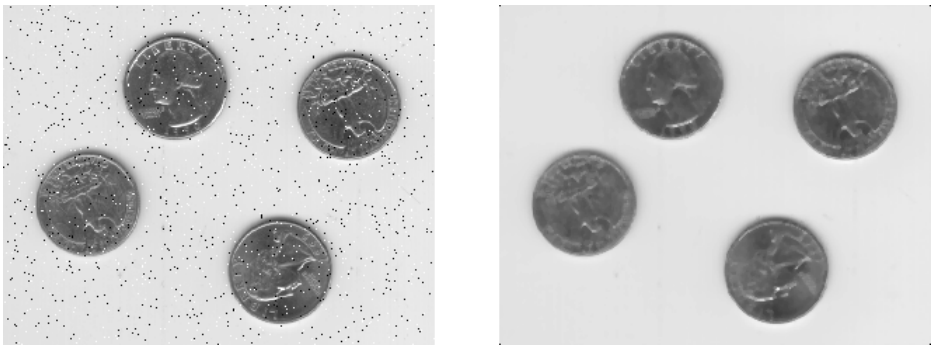


Figura 2.12 Filtro mediano

Tecniche per evidenziare i bordi

Come si può capire anche dal nome queste tecniche vengono utilizzate per evidenziare caratteristiche come i contorni o le irregolarità. Alcuni dei principali approcci consistono nello sfruttare la definizione di gradiente. In termini matematici il gradiente di una funzione $f(x, y)$ (nel nostro caso la funzione rappresenta la luminosità) è in ogni punto (x, y) dell'immagine un vettore bi-dimensionale le cui componenti sono le derivate del valore della luminosità in orizzontale e verticale:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

dove $\frac{\partial f}{\partial x}$ e $\frac{\partial f}{\partial y}$ rappresentano rispettivamente la derivata prima in orizzontale e quella in verticale. Il vettore gradiente punta nella direzione di massima velocità di variazione di f nei punti (x, y), per questo motivo ai fini della rilevazione dei bordi risulta importante l'ampiezza del vettore:

$$\nabla f = [G_x^2 + G_y^2]^{1/2}$$

che può essere approssimata nel modo seguente:

$$\nabla f \cong |G_x| + |G_y|$$

Le tecniche si differenziano a seconda del modo in cui vengono approssimate le derivate tramite i rapporti incrementali. Quelle principali sono:

- Operatore di Roberts;
- Operatore di Prewitt;
- Operatore di Sobel;

Ognuno dei tre operatori consiste nell'effettuare la convoluzione dell'immagine originale con le rispettive maschere [15]:

$$G_x = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix} \quad \text{Roberts}$$

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} \quad G_y = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad \text{Prewitt}$$

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \quad \text{Sobel}$$

Figura 2.13 Filtro di Sobel

Un'altra tecnica molto utilizzata ed efficiente per il riconoscimento dei contorni è data dall'algoritmo di Canny. Come prima operazione l'immagine viene sottoposta ad un filtro gaussiano, che introduce una leggera sfuocatura a seconda delle sue dimensioni. Successivamente vengono utilizzati quattro filtri differenti per il rilevamento dei contorni orizzontali, verticali e diagonali. Per ogni pixel si considera come valida la direzione relativo al filtro che fornisce il valore maggiore. Questa direzione, combinata con il valore del filtro, restituisce il massimo gradiente di luminosità in ogni punto dell'immagine. Per individuare i bordi si potrebbero considerare i punti dell'immagine gradiente con valori elevati. Questo approccio non risulta però sufficiente per determinare se un punto appartiene o meno ad un contorno. Una possibile soluzione consiste nell'utilizzare una finestra centrata sul pixel in elaborazione e prendere solo i pixel che soddisfano la condizione di massimo locale, cioè quei pixel la cui derivata del gradiente risulta nulla. Questa operazione prende il nome di soppressione dei non massimi. L'ultima fase di estrazione dei bordi si esegue tramite un procedimento di sogliatura: si definiscono due soglie che vengono confrontate con il gradiente in ogni punto. Se il punto è inferiore alla soglia bassa viene scartato, se il punto è superiore a quella alta viene accettato, mentre se è compreso la tre due soglie viene accettato solo se adiacente a pixel validi. Il risultato di questa ultima operazione è un'immagine binaria in cui sono evidenziati i contorni.

Figura 2.14 Metodo di Canny

2.3 Estrazione di features

L'estrazione delle caratteristiche costituisce la seconda fase di un sistema di *pattern recognition*. Questa operazione risulta essere molto importante ai fini della classificazione dell'oggetto fornito in input. L'estrazione delle *features* deve essere effettuata in modo tale da estrarre le informazioni rilevanti ai fini del riconoscimento. Esse consentono di rappresentare immagini utilizzando una ridotta quantità di informazioni, aspetto molto importante per aumentare l'efficienza degli algoritmi e diminuire di conseguenza il loro costo computazionale. Il concetto di *feature* non ha una definizione ben precisa, essa potrebbe catturare un particolare dell'immagine come la sua area, il perimetro o i bordi contenuti in essa, sostanzialmente con caratteristica si indica una parte dell'immagine utile per l'elaborazione, quindi la sua definizione dipende dal contesto che si sta affrontando. Nel campo delle elaborazioni delle immagini risultano di particolare interesse l'utilizzo di *features* invarianti. Con questo termine si indicano caratteristiche estratte dai *patterns* che sono il più possibile costanti rispetto alla traslazione, rotazione o scalatura dell'immagine [13]. Tra le *features* che soddisfano queste condizioni troviamo i momenti, che nel campo delle immagini vengono spesso utilizzati per ricavare caratteristiche che vengono poi utilizzate per effettuare la classificazione di immagini bidimensionali. Hu nel 1961 fu il primo ad introdurre un insieme di momenti invarianti utilizzando una

combinazione non lineare di momenti regolari. I momenti da lui individuati avevano l'importante proprietà di essere contemporaneamente invarianti a traslazioni, rotazioni o scalature dell'immagine. Uno dei loro principali svantaggi era dato dal fatto che la ricostruzione dell'immagine a partire da tali momenti risultava difficoltosa. Per risolvere questi problemi, Teague suggerì l'utilizzo di momenti ortogonali per recuperare l'immagine a partire dai momenti invarianti utilizzando la teoria dei polinomi ortogonali. Egli introdusse i momenti di Zernike, facilmente realizzabili anche utilizzando ordini arbitrariamente elevati. Nei paragrafi successivi introdurremo i momenti regolari, i sette momenti di Hu ed i momenti di Zernike, seguirà una breve panoramica della trasformata di Hough, tecnica utilizzata per il riconoscimento di forme all'interno di una immagine.

2.3.1 Momenti

Data una funzione continua bidimensionale $f(x, y)$ il momento (chiamato anche momento semplice o regolare) di ordine $(p + q)$ è definito come:

$$M_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y)\, dxdy$$

con $p, q = 0,1, \dots, \infty$. Nel caso delle immagini digitali, i cui pixel possiedono una intensità $f(x, y)$ è sufficiente sostituire l'integrale con la sommatoria e si ottiene:

$$M_{pq} = \sum_{x} \sum_{y} x^p y^q f(x, y)$$

In analogia alla definizione precedente il momento centrale è definito come:

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \overline{x})^p (y - \overline{y})^q f(x, y)\, dxdy$$

dove con $\bar{x} = \frac{M_{10}}{M_{00}}$ e $\bar{y} = \frac{M_{01}}{M_{00}}$ indichiamo le componenti del centroide, cioè il baricentro dell'immagine, che vengono calcolate sfruttando i momenti regolari. Lavorando con immagini la loro definizione viene modificata come segue:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y)$$

Tali momenti sono equivalenti ai momenti regolari M_{pq} di una immagine che è stata spostata in modo tale che il suo baricentro $(\bar{x}, \bar{y})$ coincida con l'origine del sistema di riferimento. Di conseguenza i momenti centrali μ_{pq} sono invarianti per traslazione. È possibile anche definire i momenti centrali normalizzati η_{pq}, con $p + q \geq 2$, che possono essere costruiti per essere invarianti sia a traslazioni che a cambiamenti di scala dividendo il corrispondente momento centrale per il momento di ordine 00 propriamente scalato:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{(1+\frac{p+q}{2})}}$$

2.3.2 Momenti di Hu

Si basano sui momenti centrali normalizzati (η_{pq}) e sono contemporaneamente invarianti a traslazioni, rotazioni e cambiamenti di scala. I sette momenti invarianti di Hu sono così definiti [7],[8]:

$$\varphi_1 = \eta_{20} + \eta_{02}$$

$$\varphi_2 = (\eta_{20} - \eta_{02})^2 + (2\eta_{11})^2$$

$$\varphi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\varphi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\varphi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$\varphi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$\varphi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

Queste *features* possono essere calcolate sia sulle immagini binarie che su quelle a livello di grigio. I momenti di Hu prendono in considerazione ogni pixel dell'immagine, e proprio per questo motivo essi hanno un elevato costo computazionale. Una possibile soluzione a questo problema (utilizzata anche in fase di sviluppo del progetto) consiste nel ridurre la risoluzione spaziale dell'immagine, concentrandosi solo sull'oggetto da analizzare, in modo da diminuire il numero di pixel su cui deve essere effettuato il calcolo.

2.3.3 Momenti di Zernike

I momenti di Zernike rappresentano un insieme di momenti, ottenuti a partire da un insieme di polinomi complessi che risultano ortogonali all'interno del cerchio unitario $\{x^2 + y^2 = 1\}$ [14]. La proprietà di ortogonalità di tali momenti consente di descrivere gli oggetti in esame utilizzando un ridotto numero di *features*. La forma dei polinomi è la seguente:

$$V_{nm}(x, y) = V_{nm}(\rho, \theta) = R_{nm}(\rho)e^{im\theta}$$

i valori n ed m sono interi che soddisfano le seguenti condizioni:

$$n \geq 0, n - |m| = pari, |m| \leq n$$

ρ rappresenta la lunghezza del vettore dall'origine fino al pixel (x, y), mentre θ è l'angolo compreso tra il vettore ρ e l'asse delle x. Infine $R_{nm}(\rho)e^{im\theta}$, detti polinomi di Zernike, rappresentano un insieme di funzioni ortogonali a valori complessi all'interno del cerchio unitario:

$$R_{nm}(\rho) = \sum_{s=0}^{(n-|m|)/2} (-1)^s \frac{(n-s)!}{s!\left(\frac{n+|m|}{2} - s\right)!\left(\frac{n-|m|}{2} - s\right)!} \rho^{n-2s}$$

I momenti di Zernike di ordine n , con un numero di ripetizioni pari ad m di una funzione continua bidimensionale $f(x, y)$ sono definiti come:

$$Z_{nm} = \frac{n+1}{\pi} \int\int_{x^2+y^2 \leq 1} f(x, y) V_{n,-m}(x, y)\ dxdy$$

Lavorando con le immagini digitali è sufficiente sostituire gli integrali con le sommatorie:

$$Z_{nm} = \frac{n+1}{\pi} \sum_x \sum_y f(x,y) V_{n,-m}(x,y), \quad x^2 + y^2 \leq 1$$

I momenti di Zernike risultano essere invarianti solo ad eventuali rotazioni dell'immagine, per ottenere l'invarianza per scala e traslazione è necessario effettuare una normalizzazione dell'immagine rispetto a questi due fattori. Supponendo di conoscere tutti i momenti di Zernike di una certa immagine $f(x,y)$ fino a quelli di ordine N è possibile ricostruirla a partire dai momenti stessi secondo il seguente modo:

$$f'(x,y) = \sum_{n=0}^{N} \sum_m Z_{nm} V_{nm}(x,y)$$

2.3.4 Momenti di Pseudo-Zernike

Questi momenti sono stati ricavati dai momenti di Zernike e di conseguenza presentano proprietà simili. L'aspetto che differenzia questi momenti da quelli di Zernike è dato dal fatto che i polinomi radiali sono definiti come:

$$R'_{nm}(\rho) = \sum_{s=0}^{n-|m|} (-1)^s \frac{(2n+1-s)!}{s!\,(n-|m|-s)!\,(n+|m|+1-s)!} \rho^{n-s}$$

I momenti vengono poi così definiti:

$$Z_{nm} = \frac{n+1}{\pi} \sum_x \sum_y f(x,y)\, R'_{nm}(\rho) e^{-im\theta}$$

i valori n ed m sono interi che soddisfano le seguenti condizioni:

$$n \geq 0, |m| \leq n$$

ρ rappresenta la lunghezza del vettore dall'origine fino al pixel (x,y), mentre θ è l'angolo compreso tra il vettore ρ e l'asse delle x. Come si può notare per questo tipo di momenti viene a decadere la condizione $n - |m| = pari$ dei momenti di Zernike. Si può anche notare che il numero dei momenti di pseudo-

Zernike risultano essere circa il doppio rispetto a quelli "originali", di conseguenza l'immagine ricostruita con questi momenti possiederà maggiori dettagli rispetto a quelli tradizionali di Zernike.

2.3.4 Trasformata di Hough

La trasformata di Hough rappresenta una tecnica di estrazione delle *features* molto utilizzata nel campo della elaborazione delle immagini. Essa consente di riconoscere particolari configurazioni di punti presenti nell'immagine, come segmenti, curve o altre forme prefissate. L'idea base di questa tecnica è quella di mappare un problema difficile di riconoscimento di forme in un problema (più facile) di rilevazione dei picchi nello spazio dei parametri della curva cercata. Se consideriamo l'equazione della retta $y = mx + c$ possiamo notare che essa è identificata dalla coppia di parametri (m, c). Nel caso delle rette, quindi, lo spazio dei parametri è formato da un piano ed esse vengono rappresentate tramite un punto. Viceversa, ogni punto nello spazio di partenza (x, y) rappresenta una retta nello spazio dei parametri.

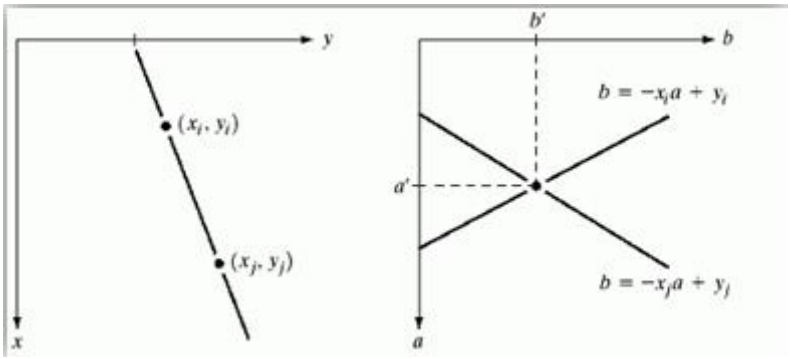


Figura 2.15 Spazio dell'immagine (x,y) e spazio dei parametri (m,c)

Come si può notare in Figura 2.15 due punti appartenenti alla stessa retta, corrispondono nello spazio dei parametri a due rette, la cui intersezione ci fornisce i parametri (m, c) della retta di partenza. Analogamente una retta nello spazio di partenza definita da N punti viene identificata, nello spazio dei parametri, come intersezione di N rette, ognuna corrispondente ad uno degli N punti. Se si possiedono numerosi punti nello spazio di partenza, questo problema

può essere ricondotto ad un problema di *edge detection* di picchi nello spazio dei parametri. L'implementazione della trasformata di Hough per il riconoscimento di forme (nel nostro caso analizzeremo rette di parametri (m, c)) può essere effettuata nel modo seguente: come prima operazione effettuiamo una quantizzazione dello spazio dei parametri, scegliendo adeguati valori di minimo e massimo per c ed m. Ad ogni cella della griglia finita così ottenuta assegniamo un contatore opportunamente inizializzato a zero. Successivamente ad ogni punto dell'immagine applichiamo un gradiente (Roberts, Prewitt, Sobel…) e se il suo valore supera una certa soglia incrementiamo di una unità il corrispondente contatore. A questo punto si utilizza una tecnica di valutazione basata su voto, ovvero i punti nello spazio dei parametri che hanno accumulato il maggior numero di voti sono quelli che hanno probabilità elevata di rappresentare curve presenti nell'immagine. Lo stesso principio può essere applicato nell'individuazione di forme di altro tipo, scegliendo come variabili opportuni parametri che ne definiscono la forma cercata.

Figura 2.16 Esempio di utilizzo della trasformata di Hough per individuazione delle linee

2.4 Classificazione

L'ultima fase di un sistema di *pattern recognition* è costituita dalla classificazione. Scopo di queste tecniche è quello di individuare la classe a cui appartiene l'oggetto da riconoscere. Ogni classe raggruppa oggetti aventi proprietà comuni. Come già descritto nel precedente capitolo per poter effettuare il riconoscimento i classificatori devono essere addestrati ed è quindi necessario utilizzare un insiemi di dati di addestramento che prende il nome di *training set*. Questo insieme viene utilizzato, come dice il termine stesso, per addestrare l'algoritmo che, in base alla classificazione fornita, è in grado di individuare le caratteristiche che discriminano gli elementi appartenenti alle differenti classi. La classificazione può essere effettuata in modo supervisionato qualora il progettista abbia definito a priori le classi di interesse, o in modo non supervisionato nel caso in cui è il sistema stesso che deve determinare in modo autonomo la classe di appartenenza del *pattern* a partire dai dati forniti in input. Uno degli algoritmi più utilizzati per la classificazione di oggetti è il *k-nearest neighbor*.

2.4.1 K-nearest neighbor

Questo algoritmo effettua la classificazione supervisionata basandosi sulle caratteristiche degli oggetti vicini a quello considerato. Il parametro k viene riportato all'interno del nome dell'algoritmo in quanto l'oggetto viene classificato basandosi sulla maggioranza dei voti dei suoi k vicini. Se k=1 allora l'oggetto viene assegnato alla classe del suo vicino, ed in questo caso si parla di *nearest neighbor* (il più vicino). La scelta del parametro k viene influenzata dalle caratteristiche dei dati e l'impiego di un elevato numero di *features* per effettuare la classificazione porta ad una diminuzione delle prestazioni. Per questo motivo si sceglie solitamente di utilizzare come numero di vicini un valore che fornisca un buon compromesso tra prestazione e classificazioni effettuate positivamente. Dato che il riconoscimento viene effettuato considerando i voti dei k oggetti vicini si potrebbe verificare il problema dovuto alla predominanza delle classi

con il maggior numero di oggetti. In questo caso una soluzione consisterebbe nel pesare i contributi dei vicini in funzione della loro distanza dall'oggetto considerato. Il funzionamento dell'algoritmo è il seguente: durante la prima fase viene effettuato l'addestramento del sistema. Ad ogni *features* viene assegnata una dimensione in modo da formare uno spazio multidimensionale delle caratteristiche. All'interno di questo spazio vengono riportate le *features* estratte da un *training set* etichettato, in cui le classi sono già state definite a priori. Ogni oggetto rappresenta un punto all'interno dello spazio multidimensionale. Completata la fase di apprendimento si iniziano ad analizzare i campioni da classificare: da ognuno di essi si estraggono le caratteristiche, che vengono poi confrontate con quelle presenti nello spazio. Il confronto viene effettuato in termini di distanze ed una delle tecniche più utilizzate consiste nel calcolo della distanza euclidea tra due punti $A(a_1, a_2, \dots, a_n)$ e $B(b_1, b_2, \dots, b_n)$:

$$d(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Un punto, che nello spazio multidimensionale rappresenta un oggetto, viene assegnato ad una classe se questa è quella più votata tra i k campioni più vicini all'oggetto in esame.

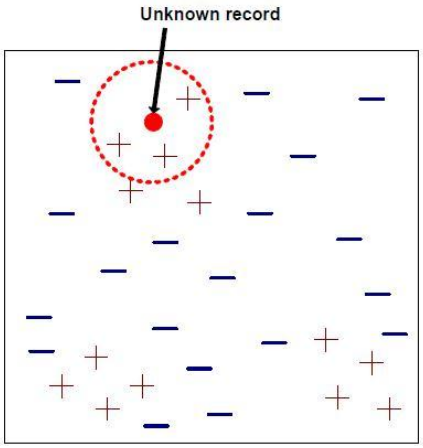


Figura 2.17 Esempio di applicazione del k-nearest neighbor con k=3

Questi tipi di classificatori sono chiamati anche *instance-based* o *lazy learner* poiché immagazzinano tutti i campioni dell'insieme di addestramento e ritardano la costruzione del modello fino alla classificazione di un nuovo campione. Questo metodo si contrappone a quello di tipo *eager learning* (esempio reti neurali) che prova a costruire un modello generalizzato prima di ricevere i nuovi campioni. Il principale vantaggio di queste tecniche è dato dal fatto che durante la fase di addestramento si effettua un'approssimazione globale, di conseguenza lo spazio richiesto è molto inferiore rispetto alle tecniche *instance-based*. Questi ultimi metodi possono richiedere costi computazionali eccessivi quando il numero di potenziali vicini con cui bisogna confrontare l'oggetto in input risulta essere grande. Essi risultano essere più veloci nella fase di apprendimento, in quanto ritardano la costruzione del modello, ma più lenti nella classificazione rispetto a quelli *eager learning* a causa dei calcoli che vengono effettuati.

Capitolo 3

Sviluppo dell'applicazione

Di seguito verrà fornita una introduzione del dominio applicativo in cui è collocato il progetto, in cui si discutono i problemi affrontati e le idee utilizzate per la realizzazione. Seguirà una descrizione esaustiva dell'implementazione delle tre sottofasi principali che costituiscono il progetto.

3.1 Introduzione al problema

Lo scopo del progetto è quello di realizzare un'applicazione in Matlab in grado di effettuare il riconoscimento automatico della data (anno) di prossimo collaudo nelle bombole a gas. Questa idea nasce da una richiesta reale da parte di un'azienda di realizzare un impianto automatico in grado di effettuare il riconoscimento. L'utilizzo di un sistema di questo tipo, porta infatti importanti benefici: non sono necessari operai, ma il tutto viene delegato a macchinari predisposti, inoltre si ha un notevole incremento del numero di bombole che vengono esaminate, in quanto le macchine effettuano il riconoscimento in maniera molto più veloce di quanto effettuato da umani. Un fattore fondamentale di questo secondo aspetto è che il tutto sia realizzato *real time,* ovvero in maniera pressoché istantanea, altrimenti verrebbero a decadere i vantaggi portati dall'utilizzo di sistemi automatici. Per rispettare questo requisito si è deciso di utilizzare algoritmi con costi computazionali non elevati, in modo tale che il sistema potesse fornire risposta in breve termine. L'idea di base è la seguente: le bombole che devono essere esaminate vengono collocate sopra un nastro trasportatore che le ordina in fila indiana. Ad un certo punto del nastro è collocato un sensore, che si occupa di effettuare l'acquisizione della parte superiore della bombola. In questa posizione viene infatti inserita l'etichetta (di colore giallo) contenente la data di prossimo collaudo che dovrà essere analizzata

dal sistema. L'immagine acquisita viene poi inviata ad un calcolatore, su cui è presente l'applicativo, che avrà il compito di classificarla. Il programma confronterà la data riconosciuta dall'etichetta con la data attuale, e nel caso in cui la prima superi la seconda, la bombola verrà classificata come "da collaudare" ed in un punto successivo del nastro verrà inserita fra quelle che non hanno superato positivamente il test.

Per la realizzazione dell'applicazione è stato adottato lo schema di un generico sistema di *pattern recognition* composto dai moduli di pre-elaborazione, estrazione delle *features* e classificazione. Prima di iniziare l'implementazione è stata effettuata un' analisi del problema, per definire le strategie da adottare. Le immagini che vengono acquisite dal sensore non presentano tutte la stessa qualità, ma alcune sono più chiare o più scure ed inoltre, per alcune di esse, l'etichetta contenente la data risulta essere meno pulita, rendendo il riconoscimento ancora più difficoltoso (vedi Figura 3.1).

a)

b)

Figura 3.1 Esempi delle diverse qualità delle immagini: a) immagine di scarsa qualità e b) immagine di buona qualità

Per tutti questi motivi è necessaria una fase di pre-elaborazione dell'immagine, in cui vengono rimossi il più possibile i difetti contenuti in essa, come ad esempio il rumore introdotto in fase di acquisizione. Completata questa procedura si inizia la fase di riconoscimento della data. Questa è stata suddivisa in 3 sottofasi principali:

- Individuazione dell'etichetta;
- Individuazione della data;
- Riconoscimento della data;

Le prime due vengono utilizzate per avvicinarsi gradualmente alla data, fino a posizionarsi correttamente su di essa. Infatti il sensore che effettua l'acquisizione possiede una risoluzione di 1624x1234 che potrebbe creare problemi nel garantire una valutazione in tempo reale della data. Proprio per questo motivo si è deciso di utilizzare queste due fasi: la prima ha il compito di individuare l'etichetta di colore giallo contenente la data, in questo modo le immagini vengono ridotte dalla risoluzione originale ad una risoluzione di circa 1000x1000, mentre la seconda si occupa di individuare il pittogramma stampato sull'etichetta. Da questa posizione ci si sposterà in maniera diametralmente opposta per collocarsi sulla data di collaudo. Al termine di questa operazione si ottiene un'immagine con una risoluzione di circa 400x400, che risulta essere molto più piccola rispetto a quella di partenza. L'ultima sottofase costituisce il cuore dell'applicativo e si occupa di effettuare la classificazione della data contenuta nell'immagine restituita al passo precedente.

Il sistema sviluppato risulta essere completamente automatico, tuttavia per poter mostrare i risultati ottenuti ad ogni passo si è deciso di sviluppare una interfaccia differente per ognuna delle tre sottofasi considerate. In ognuna delle tre interfacce è l'utente stesso che avvia la computazione tramite clic su un pulsante, ma l'applicazione, se necessario, può essere completamente automatizzata per restituire il riconoscimento della data senza alcuna interazione da parte dell'utente. Come risultato finale viene mostrata la data di prossimo collaudo che è stata riconosciuta. Le varie interfacce che verranno mostrate durante questo

capitolo sono molto semplici. Nelle applicazioni di elaborazione delle immagini, infatti, la complessità è concentrata nello sviluppo e nella scelta di quali algoritmi e tecniche utilizzare per raggiungere l'obiettivo preposto. L'interfaccia ha l'unico scopo di mostrare i risultati raggiunti applicando determinati algoritmi, trasparenti all'utente, all'immagine di partenza. All'avvio del programma viene mostrata la seguente form:

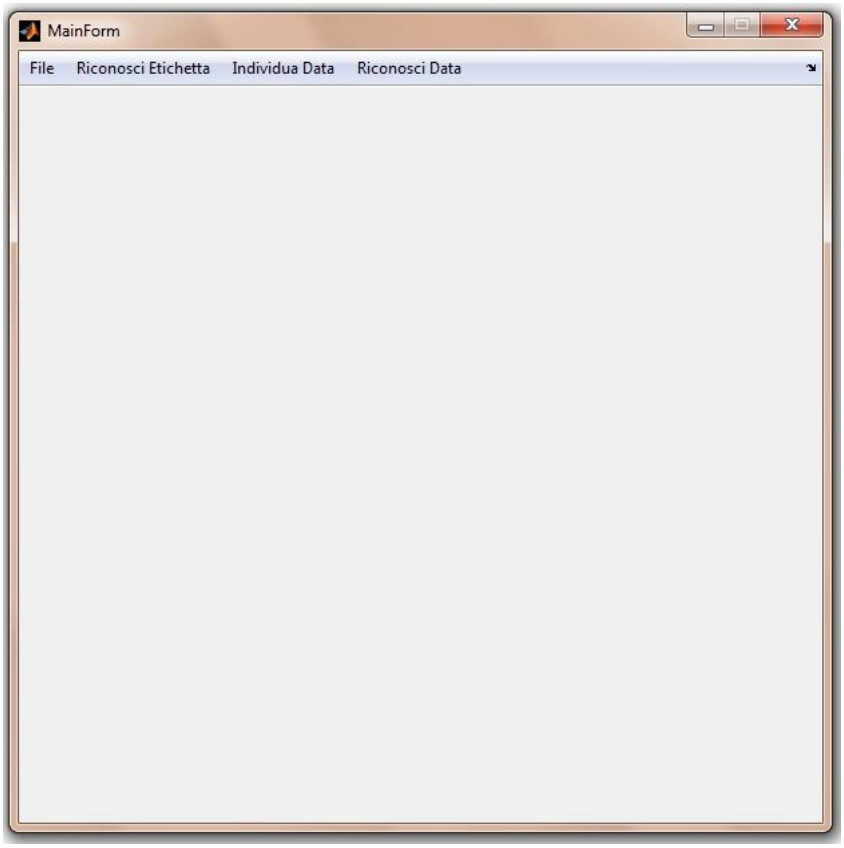


Figura 3.2 Form di avvio dell'applicazione

In seguito verranno descritte le implementazioni delle tre sottofasi principali che costituiscono l'applicazione. Al termine di ognuna di esse verrà riportato lo pseudo-codice che fornirà una descrizione, in maniera semplicistica, di come è stata realizzata l'implementazione. L'immagine che verrà utilizzata per mostrare il funzionamento dell'applicazione è la seguente:

Figura 3.3 Immagine utilizzata come esempio

3.2 Localizzazione dell'etichetta

Questa fase rappresenta il primo passo di avvicinamento alla data da riconoscere. Essa ha il compito di localizzare l'etichetta tonda di colore giallo. Al suo interno è stata anche inserita la funzione di pre-elaborazione, che come già anticipato avrà il compito di migliorare la qualità dell'immagine in input. La principale operazione svolta dalla fase di *preprocessing* è la riduzione del rumore che viene introdotto in fase di acquisizione. Questa viene effettuata applicando un filtro mediano di dimensione 3x3 su ognuno dei tre canali RGB che compongono l'immagine. Completate queste operazioni preliminari si procede all'individuazione dell'etichetta. Come si può notare dalla Figura 3.3 essa possiede una forte predominanza del colore giallo, per questo motivo si è deciso di sfruttare l'intensità luminosa dei pixel gialli per effettuare l'individuazione. La prima operazione consiste nell'effettuare un *tresholding* dell'immagine, con la stessa risoluzione di quella di partenza. Il risultato è una semplice immagine binaria in cui i pixel bianchi sono quelli che hanno superato il valore di soglia, mentre i neri quelli che non l'hanno superato. Come soglia è stato utilizzato un preciso valore di giallo, leggermente più scuro rispetto a quello presente

sull'etichetta, in modo da prelevare anche eventuali altri pixel che risultano essere più scuri a causa di ombre.

Figura 3.4 Immagine ottenuta dopo il tresholding

Come si può notare dalla Figura 3.4 grazie al *tresholding* viene messa bene in evidenza la parte interessata, che verrà ora ritagliata in modo da diminuire il numero di pixel dell'immagine da esaminare. Prima di effettuare questa operazione è pero necessario applicare una seconda volta un filtro mediano, in questo caso di dimensione 5x5, che ha lo scopo di rimuovere eventuali pixel che sono stati selezionati, ma non appartengono all'etichetta. La fase di taglio dell'immagine viene effettuata cercando, all'interno dell'immagine binarizzata, i pixel posti ad 1 (pixel bianchi) che presentano riga minima, massima e colonna minima, massima. Questi rappresenteranno i bordi della nuova immagine, che viene estratta da quella di partenza tramite la funzione *imcrop*. L'utilizzo del filtro mediano risulta quindi essere molto importante, perché altrimenti potrebbero essere selezionati dei pixel non appartenenti all'etichetta che non consentirebbero la corretta applicazione del *crop*. Lo pseudo-codice, che descrive l'implementazione della procedura, viene riportato qui di seguito:

```
Inizio

    - Applicazione del filtro mediano 3x3 per ogni canale RGB
      dell'immagine;
    - Tresholding dell'immagine basato su tonalità di giallo per la
      localizzazione dell'etichetta;
    - Applicazione del filtro mediano 5x5 per rimuovere eventuali
      pixel non appartenenti all'etichetta;
    - Localizzazione dell'etichetta tramite individuazione di riga
      minima, massima e colonna minima, massima per i pixel bianchi
      dell'immagine binaria;
    - Estrazione dell'etichetta tramite utilizzo degli estremi appena
      Individuati;

Fine
```

Al termine di questa operazione l'interfaccia che implementa questa prima sottofase mostra il seguente risultato:

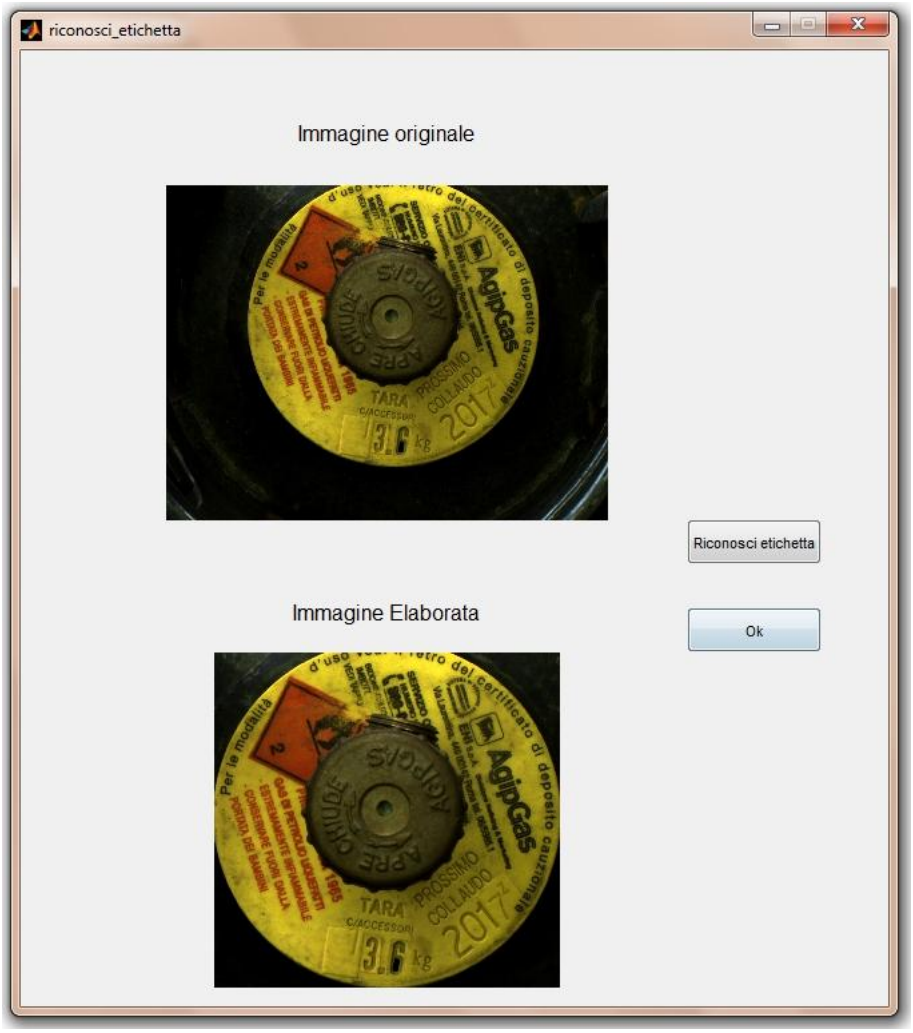


Figura 3.5 Localizzazione dell'etichetta

3.3 Individuazione della data

Nella seconda sottofase si procede all'individuazione della data a partire dall'immagine fornita in output al passo precedente. Questa viene realizzata sfruttando il pittogramma contenuto nell'etichetta. Esso, infatti, risulta essere diametralmente opposto alla data, di conseguenza per posizionarsi in maniera corretta sarà sufficiente individuare tale marchio. Per la sua localizzazione, a seconda dell'immagine, è stata utilizzata una tra queste due tecniche principali:

- Riconoscimento del colore del pittogramma;
- Trasformata di Hough per l'individuazione dei lati del pittogramma;

La prima si basa sulla stessa idea impiegata per effettuare la localizzazione dell'etichetta, cioè sfruttare l'intensità luminosa dei pixel. Anche in questo caso, infatti, si effettua una operazione di *tresholding* che produce una immagine binaria in cui i pixel che sono posti ad 1 sono quelli che superano il valore di soglia, definito utilizzando una particolare tonalità di rosso (colore predominante del pittogramma). In seguito si applica il filtro mediano per rimuovere eventuali pixel non appartenenti alla parte interessata. La singola operazione di *tresholding* non risulta però sufficiente, in questo caso nell'immagine sono presenti delle scritte rappresentanti avvertenze che presentano lo stesso colore del pittogramma e che vengono anch'esse selezionate (vedi Figura 3.6).

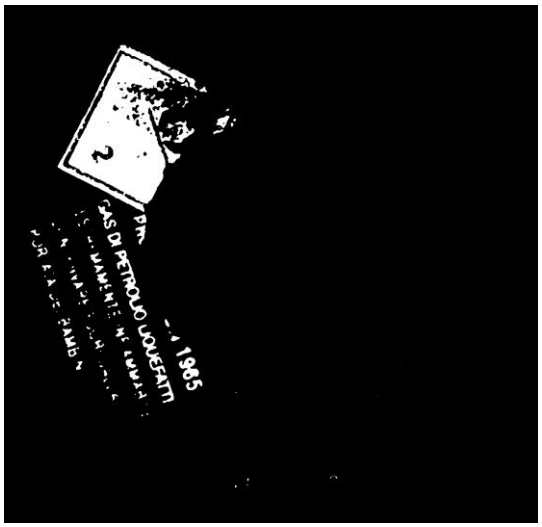


Figura 3.6 Immagine dopo il tresholding

Per questo motivo, dopo la binarizzazione, vengono individuate le componenti connesse dell'immagine e viene analizzata l'area di ogni singolo elemento. Come si può vedere in Figura 3.6 l'area della componente rappresentante il pittogramma risulta essere quella maggiore, di conseguenza è necessario effettuare un ulteriore filtraggio che mantiene solamente l'oggetto avente la massima area. A questo punto, come per la procedura di localizzazione dell'etichetta, si selezionano i pixel aventi riga minima, massima e colonna minima, massima che costituiranno i bordi del pittogramma. Questa prima tecnica di individuazione del marchio, non risulta essere sempre applicabile, in quanto, essendo stampato sull'etichetta, in alcuni casi il pittogramma risulta avere un colore che non rientra all'interno del range di rosso considerato, perché troppo chiaro o troppo scuro. A tale scopo è stata sviluppata una seconda tecnica che, in caso di fallimento della prima, è in grado di effettuare correttamente la localizzazione. Essa sfrutta la trasformata di Hough, che risulta essere interessante per la sua particolare caratteristica di individuare eventuali linee all'interno dell'immagine in esame. I bordi che caratterizzano il pittogramma, infatti, sono ben evidenti, in quanto di colore nero su sfondo rosso, e risulta quindi semplice effettuarne l'estrazione. L'idea di base è la seguente: inizialmente si individuano i contorni dell'intera immagine applicando l'operatore di Sobel. Successivamente si calcola la trasformata tramite la funzione *hough* presente in MatLab che inserisce il risultato in una matrice che chiameremo matrice di Hough. Tramite la funzione *houghpeaks* vengono prelevati i tre valori più elevati della matrice. Ricordando la teoria di Hough (Paragrafo 2.3.4) le celle che presentano valore più alto sono quelle che hanno la maggiore probabilità di corrispondere a rette. Si è deciso di prelevare i prime tre valori più elevati perché è stato studiato, in modo empirico effettuando numerose prove su differenti immagini, che risulta essere un numero sufficiente per individuare almeno due dei quattro lati del pittogramma. Se si osserva, infatti, l'immagine utilizzata in questo esempio (Figura 3.3), si può notare che le linee più evidenti sono proprio quelle che costituiscono i bordi del marchio, esse

avranno quindi elevata probabilità di essere estratte tramite l'utilizzo della trasformata di Hough. Per completare l'individuazione dei bordi è stata utilizzata la funzione *houghlines* che, a partire dall'immagine binaria contenente i contorni e dai tre valori prelevati in precedenza dalla matrice di Hough, restituisce le coordinate del primo ed ultimo punto di ognuno dei segmenti individuati. Di ogni linea viene poi calcolata la lunghezza ed in seguito viene utilizzata per l'estrazione del marchio quella più lunga che rientra all'interno di un ristretto intervallo corrispondente alla lunghezza del lato del pittogramma. In questo modo, viene localizzato un solo lato del pittogramma. Per individuare i due lati adiacenti, è stato utilizzato il concetto di perpendicolarità. Essendo il marchio un quadrato, tali segmenti sono effettivamente perpendicolari a quello appena individuato e presentano anche la stessa lunghezza, di conseguenza è sufficiente sfruttare questa proprietà per effettuarne la localizzazione. Una volta individuati tre dei quattro lati si utilizzano i loro punti estremi per effettuare l'estrazione del pittogramma. Completata l'identificazione ci si sposta in posizione diametralmente opposta per collocarsi sulla data di prossimo collaudo. Lo pseudo-codice che fornisce una descrizione di come è stata realizzata questa fase viene riportato qui di seguito:

```
Inizio

    - Tresholding dell'immagine basato su tonalità di rosso per la
      localizzazione del pittogramma;
    - Applicazione del filtro mediano 5x5 per rimuovere eventuali
      pixel non appartenenti al pittogramma;
    - Individuazione delle componenti connesse che possiedono area
      sopra una certa soglia;
    - Se non sono state individuate componenti connesse:
        - Si applica la trasformata di Hough per il riconoscimento di
          almeno due dei quattro lati del pittogramma;
        - Si identificano i restanti lati del marchio perpendicolari a
          quelli appena individuati;
        - Si utilizzano i punti estremi dei lati per la localizzazione
          del marchio;
    - Altrimenti se le componenti sono state individuate:
        - Si mantiene solamente la componente di area maggiore,
          corrispondente al marchio interessato;
        - Si effettua la localizzazione del pittogramma tramite
          individuazione di riga minima, massima e colonna minima,
          massima per i pixel bianchi dell'immagine binaria;
```

```
    - Estrazione della data tramite spostamento in posizione
      diametralmente opposta a quella del marchio;

Fine
```

Al termine di questa seconda sottofase, ottenuta utilizzando uno dei due metodi appena descritti, l'interfaccia mostra il seguente risultato:

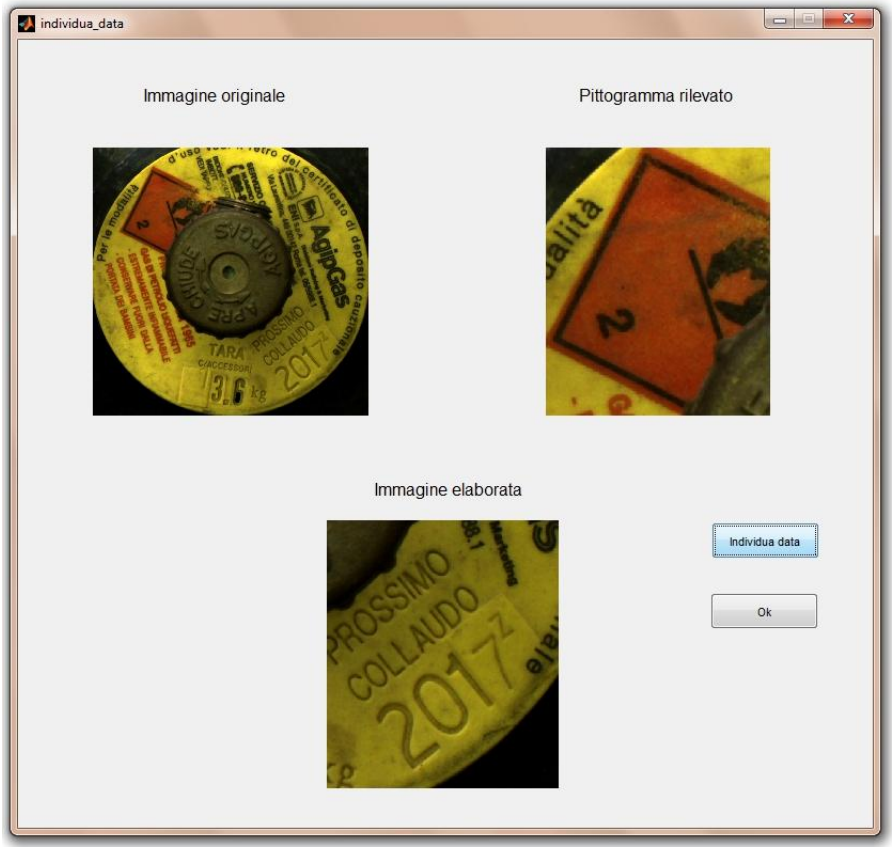


Figura 3.7 Individuazione della data

3.4 Riconoscimento della data

L'ultima delle tre sottofasi consiste nell'effettuare il riconoscimento della data impressa sull'etichetta. Come prima operazione è stata applicata una funzione *imadjust* su ognuno dei tre canali RGB che compongono l'immagine. Questa funzione modifica l'intensità dell'immagine creandone una nuova dove l'1 % dei dati è saturato alle basse ed alte intensità, come risultato si ottiene un aumento del contrasto. Successivamente l'immagine viene binarizzata, utilizzando in

questo caso un *tresholding* di tipo adattivo. Nelle precedenti sottofasi, infatti, veniva effettuata una sogliatura globale, così chiamata perché applicata su ogni pixel dell'immagine. Essa consiste nell'utilizzare un valore di soglia T ed assegnare ogni pixel di intensità maggiore a T alla regione degli oggetti e quelli inferiori alla regione dello sfondo:

$$J[x,y] = \begin{cases} 1 & se\ I[x,y] > T \\ 0 & se\ I[x,y] \leq T \end{cases}$$

La sogliatura adattiva consiste nell'applicare soglie diverse a differenti parti dell'immagine sulla base di informazioni globali e locali. Un possibile approccio per trovare la soglia locale è quello di esaminare statisticamente i valori di intensità nelle zone locali di ogni pixel. La statistica più adatta dipende fortemente dall'immagine in ingresso. Alcune delle funzioni più semplici e veloci consistono nell'utilizzare la media o il mediano dei pixel contenuti nella zona. La dimensione della finestra deve essere abbastanza grande in modo da coprire un numero sufficiente di pixel appartenenti e non allo sfondo, in caso contrario verrebbe scelto un valore di soglia troppo basso. Per lo sviluppo dell'applicazione è stata proprio utilizzata questa idea, effettuando una sogliatura locale, che sfrutta la funzione di media, con una finestra di dimensione 50.

Figura 3.8 Tresholding adattivo della data con soglia locale

L’immagine viene poi filtrata per eliminare le componenti non corrispondenti a numeri. L’operazione è realizzata analizzando l’area di ogni oggetto ed eliminando gli oggetti che non possiedono tale valore all’interno di una certa soglia, ottenuta in modo empirico esaminando l’area delle cifre che costituiscono la data, presenti nelle differenti immagini. Questa procedura elimina la maggior parte delle componenti non corrispondenti a numeri, ma una piccola parte (lettere o altre parti dell’etichetta), che possiede un’area all’interno del range stabilito, viene comunque mantenuta, per questo motivo successivamente sarà necessario effettuare ulteriori controlli per evitare di scambiare cifre con altre parti dell’immagine e viceversa. Prima di iniziare la valutazione della data si calcolano, tramite la funzione *calcolahu*, i momenti invarianti di Hu dei template creati in precedenza. La funzione crea una matrice in cui, per ogni riga, vengono inseriti i momenti dei numeri in ordine crescente. Ad ogni template corrisponde l’immagine di una delle dieci cifre possibili.

Figura 3.9 Esempio di template del numero 2

Si utilizzano i momenti invarianti di Hu, in quanto risultano contemporaneamente invarianti a traslazioni, rotazioni o scalature dell’immagine (vedi Paragrafo 2.3.2). La successiva fase consiste nel ricercare, tra tutti gli oggetti che sono stati mantenuti anche dopo il filtraggio basato su area, le prime due cifre comuni tra tutte le date, il due e lo zero. Esse risultano essere del tipo 2008, 2010, 2012… di conseguenza per individuare la data, sarà sufficiente ricercare le due cifre citate. L’idea di base è quella di calcolare i momenti invarianti degli oggetti e di individuare quali di questi vengono classificati, tramite l’algoritmo *k-nearest neighbor* (vedi Paragrafo 2.4.1), come il numero due. L’algoritmo *knn* non fa altro che confrontare i momenti di Hu dell’oggetto

in esame con la matrice contenente quelli estratti dai template e restituisce la classe (il numero) a cui i momenti si avvicinano maggiormente. Come metrica di distanza si utilizza quella euclidea. Visto che nell'immagine sono presenti altri oggetti oltre alle cifre che costituiscono la data, alcuni di essi, come ad esempio la lettera "Z", potrebbero essere classificati in maniera errata come un due. Proprio per questo motivo, i momenti invarianti delle componenti che vengono riconosciute come numero due, vengono inseriti all'interno di un vettore. Esaminati tutti gli oggetti si calcola, per tutti quelli che sono stati inseriti nel vettore, quali tra di questi assomiglia maggiormente al numero interessato. La somiglianza viene calcolata effettuando la differenza in valore assoluto tra i momenti invarianti del template corrispondente al numero due e ognuno di quelli contenuti all'interno del vettore. L'oggetto che presenta la massima somiglianza sarà quello che avrà il maggior numero di momenti (in totale quelli di Hu sono sette) che risultano i più vicini a quelli del template. Ovviamente questo sarà proprio l'oggetto che corrisponderà al numero due. Nel caso in cui non venga individuata alcuna componente dell'immagine corrispondente ad un due, si effettueranno le stesse operazioni per ricercare il numero zero, che rappresenta la seconda cifra sempre presente nelle date. Si è potuto constatare che questa tecnica risulta essere sufficientemente robusta per raggiungere un buon fattore di riconoscimento, ad eccezione di alcune immagini che risultano di scarsa qualità o per etichette eccessivamente degradate. Completata questa fase di individuazione della data si ricercano le altre cifre da classificare. Nel caso in cui sia stato individuato il primo numero della data (il numero due) viene calcolato il suo baricentro e si ricerca l'oggetto nell'immagine che possiede centroide più vicino. Essendo i numeri l'uno a fianco dell'altro la componente che verrà individuata corrisponderà alla seconda cifra della data, e sarà classificata confrontando i suoi momenti invarianti con quelli dei template tramite l'algoritmo *knn*. La procedura, a partire dal calcolo del baricentro, viene ripetuta per individuare i restanti due numeri dell'anno, considerando il centroide dell'ultima cifra classificata. Se, invece, viene rilevato come primo numero lo zero allora la procedura viene

ripetuta solamente due volte per effettuare il riconoscimento degli ultimi due numeri che compongono l'anno. Al termine della classificazione, vengono effettuati ulteriori controlli che hanno lo scopo di risolvere eventuali errori di riconoscimento (Es. sei che viene classificato come nove e viceversa). Questi controlli sono realizzati utilizzando nuovamente la differenza tra i momenti in valore assoluto. Conclusa quest'ultima operazione le quattro cifre costituenti la data vengono concatenate nel corretto ordine e stampate a video. Lo pseudo-codice relativo è il seguente:

```
Inizio

    - Regolazione dell'intensità luminosa dell'immagine;
    - Tresholding di tipo adattivo;
    - Rimozione delle componenti dell'immagine con area non contenuta
      all'interno di una soglia fissata;
    - Calcolo dei momenti di Hu per i template. Questi verranno
      confrontati con i momenti dell'oggetto in esame tramite
      l'algoritmo knn;
    - Ricerca all'interno dell'immagine binaria delle componenti, che
      tramite applicazione del knn, vengono classificate come un 2 o
      uno 0;
    - Se delle componenti sono state classificate come un 2:
        - Si individua quella che possiede i momenti che si avvicinano
          maggiormente (tramite differenza in valore assoluto) a
          quelli del template corrispondente al numero 2;
    - Altrimenti se non sono state individuate componenti classificate
      come un 2, ma sono state identificate componenti riconosciute
      come uno 0:
        - Si individua quella che possiede i momenti che si avvicinano
          maggiormente (tramite differenza in valore assoluto) a
          quelli del template corrispondente al numero 0;
    - Se la componente appena individuata risulta essere un 2:
        - Si ripetono le seguenti operazioni per 3 volte:
            - Individuazione dell'oggetto più vicino all'ultimo
              esaminato;
            - Calcolo dei suoi momenti invarianti;
            - Classificazione dell'oggetto tramite applicazione del
              knn;
    - Altrimenti, se la componente appena individuata risulta essere
      uno 0:
        - Si ripetono le seguenti operazioni per 2 volte:
            - Individuazione dell'oggetto più vicino all'ultimo
              esaminato;
            - Calcolo dei suoi momenti invarianti;
            - Classificazione dell'oggetto tramite applicazione del
              knn;
    - Si effettuano ulteriori controlli di matching per evitare errori
      nella classificazione di numeri simili (Es. 6 e 9 oppure 1 e 7);
    - Concatenazione dei quattro numeri costituenti la data;

Fine
```

La form mostra il seguente risultato:

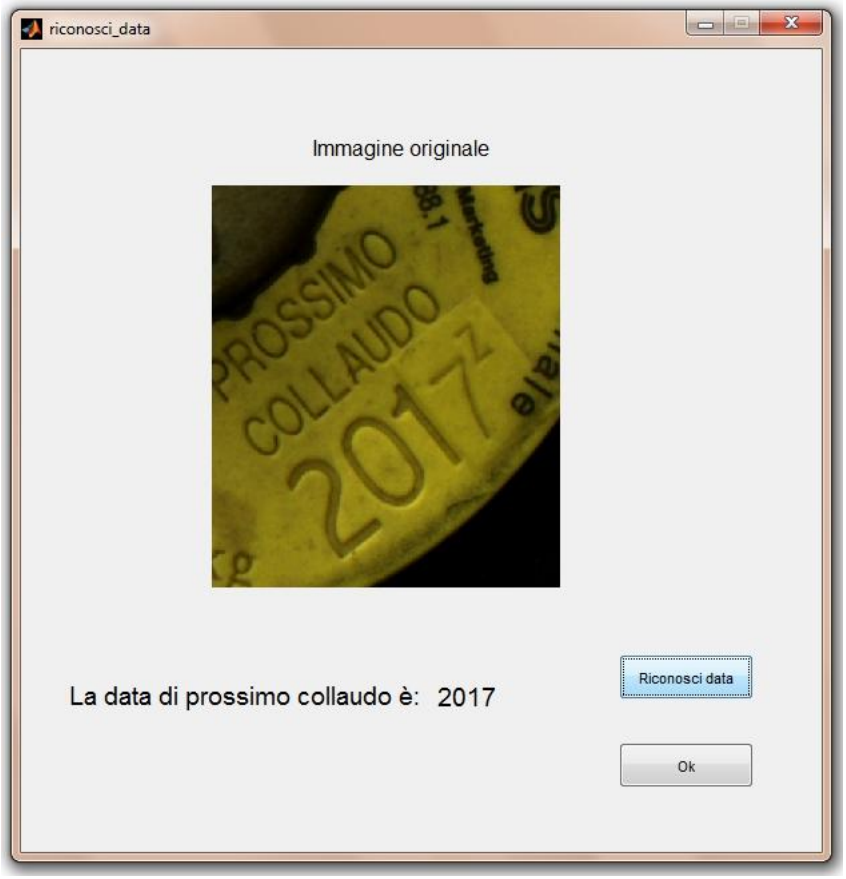


Figura 3.10 Riconoscimento della data

Capitolo 4

Analisi dei test

Nell'ultimo capitolo della tesi viene effettuata un'analisi delle prestazioni e dell'efficienza del software realizzato.

4.1 Tempi di esecuzione

Durante la fase di test sono state effettuate prove di funzionamento dell'applicazione. Nei capitoli precedenti si è parlato della necessita di fornire i risultati delle elaborazioni in tempo reale, rispetto ai tempi di esecuzione del sistema. Il vincolo più importante diventa così il tempo di risposta del sistema all'arrivo di una nuova immagine. Bisogna ricordare che il *software* è stato realizzato in MatLab, che ha il principale svantaggio di utilizzare un linguaggio interpretato e non compilato, per tale motivo i tempi di esecuzione risultano essere più alti rispetto a quelli di un compilatore C che risulta essere un linguaggio compilato. Sono stati svolti dei test per esaminare il tempo di elaborazione delle tre sottofasi principali che costituiscono l'applicazione:

Algoritmo	**Tempo di elaborazione medio (in secondi)**
Localizzazione etichetta	0.41
Individuazione data	0.72
Riconoscimento data	0.95
Tempo totale	2.08

Come si può notare dai risultati della tabella, il tempo totale medio di elaborazione risulta essere intorno ai due secondi. Il tempo di esecuzione di ognuna delle sottofasi cresce all'avvicinarsi dell'etichetta, questo perché aumenta la complessità delle operazioni svolte. Infatti il tempo maggiore viene impiegato durante il riconoscimento della data, procedura più laboriosa di tutta

l'applicazione. Il tempo risultante dalle tre fasi, seppure essendo abbastanza basso, non può essere considerato proprio a *real time*. Ogni classificazione di una immagine richiederebbe infatti all'incirca due secondi, di conseguenza si noterebbe anche ad occhio nudo il ritardo dell'elaborazione. Tuttavia nella tabella sopra riportata vengono esaminati i tempi di esecuzione delle tre sottofasi comprensivi anche delle operazioni preliminari, e quindi non facenti parte degli algoritmi utilizzati per la realizzazione dell'applicazione. L'operazione di *tresholding*, per esempio, è presente in ognuna delle fasi implementate. Essa richiede un discreto quantitativo di tempo, in quanto deve essere effettuata, ad ogni esecuzione, una valutazione dell'intera immagine. Eliminando tutte le operazioni preliminari si ottengono i seguenti tempi di esecuzione:

Algoritmo	**Tempo di elaborazione medio (in secondi)**
Localizzazione etichetta	0.04
Individuazione data	0.02
Riconoscimento data	0.50
Tempo totale	0.56

I tempi di elaborazione delle diverse sottofasi calano drasticamente e si ottiene un tempo totale medio di esecuzione di circa mezzo secondo. Da questi dati si può osservare come le operazioni preliminari risultino avere un peso determinante all'interno dell'intera computazione. La fase che presenta la minor differenza con i tempi calcolati nella tabella precedente è la terza. La motivazione è data dal fatto che in quest'ultima fase si ottiene una immagine di dimensioni ridotte, contenente solo la data da esaminare, di conseguenza l'operazione di sogliatura non risulta essere particolarmente onerosa, ma è proprio l'algoritmo utilizzato che impiega la maggior parte del tempo per essere eseguito a causa della complessità delle operazioni che devono essere svolte per effettuare la classificazione.

4.2 Principali problemi di classificazione

Non tutte le date delle immagini esaminate sono state riconosciute con successo. I principali problemi riguardanti le immagini non classificate correttamente possono essere raggruppati in due macro categorie:

- Data non leggibile;
- Etichetta non leggibile;

Di seguito verranno analizzate le due differenti problematiche.

4.2.1 Data non leggibile

Questo primo problema è causato dal fatto che la data riportata sull'etichetta non risulta essere ben impressa. L'ultimo numero (in alcune immagini anche gli ultimi due), infatti, viene riportato a mano dall'operatore e non sempre la marchiatura risulta essere effettuata nel modo corretto.

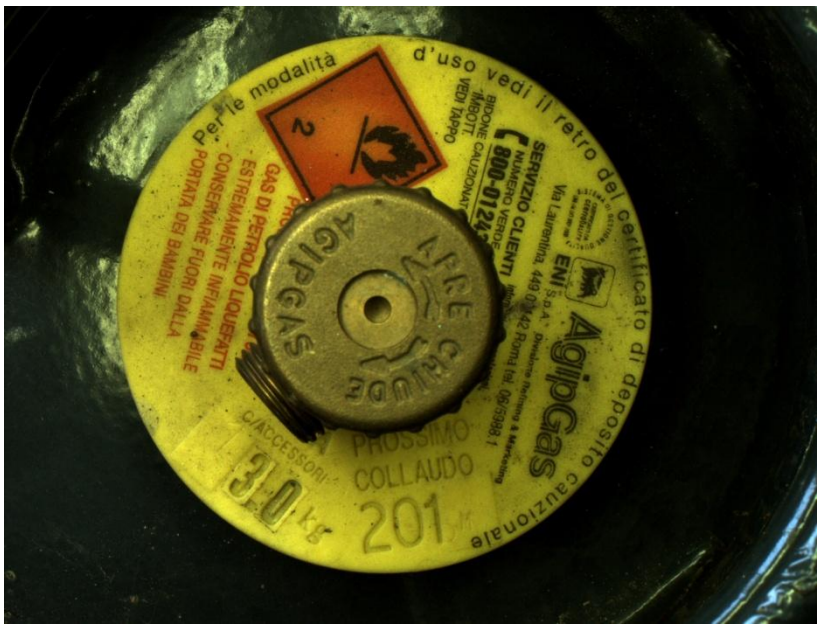


Figura 4.1 Esempio di data non leggibile

Come si può notare dalla Figura 4.1 il quarto numero della data non è stato ben riportato, ed è solo leggermente visibile. Di conseguenza l'operazione di sogliatura adattiva non sarà in grado di scegliere il valore di soglia corretto ed il numero non verrà rilevato dalla fase di *tresholding*. Di questa categoria fanno parte anche quelle immagini che contengono date con numeri sovrapposti:

Figura 4.2 Immagine contenente data con numeri sovrapposti

In questo caso è possibile individuare la data, ma non è possibile separare i numeri per effettuare la classificazione tramite i momenti invarianti di Hu.

4.2.2 Etichetta non leggibile

Questa problematica comprende tutte quelle immagini contenenti etichette che risultano essere degradate e che non consentono di effettuarne la localizzazione o individuare il pittogramma. La degradazione solitamente viene causata da agenti esterni e dipende da dove viene effettuata la collocazione della bombola. Ovviamente in caso di utilizzo esterno l'etichetta subirà una erosione causata da agenti atmosferici.

Figura 4.3 Etichetta non leggibile

Nel caso dell'immagine riportata in Figura 4.3 non risulta possibile localizzare l'etichetta. La parte superiore, infatti, viene correttamente individuata sfruttando il *tresholding*, ma non è possibile individuare il restante contorno a causa di residui solidi presenti sopra l'etichetta. Lo stesso problema si è verificato in altre immagini durante l'individuazione del pittogramma.

4.2.3 Prestazioni di riconoscimento

L'applicazione sviluppata consiste in un sistema di riconoscimento/ classificazione di oggetti. Per questo tipo di sistemi le prestazioni vengono misurate esaminando gli errori di classificazione, ovvero la percentuale dei casi in cui gli oggetti, nel nostro caso i numeri che costituiscono la data, vengono assegnati ad una classe sbagliata. Durante l'analisi dei test, sono quindi stati esaminati i casi in cui il riconoscimento non è andato a buon fine. Oltre alle problematiche descritte in precedenza, che non hanno consentito di effettuare la fase finale di classificazione della data, si sono verificati anche errori nel riconoscimento. Questi sono dovuti principalmente a due motivi:

- Errore nella classificazione di numeri strutturalmente simili;
- Errore nella classificazione a causa della non perfetta sogliatura;

Nel primo caso il numero viene scambiato con un altro, non corretto. Questo errore si verifica tra numeri che presentano una struttura simile anche in caso di rotazioni, scalature o traslazioni (Es. numeri 7-1 o 6-9). Per tali numeri sono stati utilizzati nell'applicazione ulteriori controlli che riducessero al minimo gli errori di classificazione di questo tipo, ma alcuni casi anomali si sono comunque verificati. Di seguito vengono riportate alcune immagini che presentano questo problema:

Figura 4.4 Immagini con errori di classificazione dovuti a numeri simili

Nel secondo caso gli errori di riconoscimento sono dovuti all'operazione di sogliatura. Dopo il *tresholding* e la conseguente binarizzazione si esegue una operazione di pulitura che ha il compito di rimuovere le parti meno significative. Per alcuni tipi di immagini l'operazione di sogliatura è in grado di rilevare correttamente i numeri, tuttavia essi sono formati da più componenti non connesse tra loro che vengono successivamente rimosse dall'operazione di filtraggio. A questo punto il numero da classificare viene privato di una sua parte significativa che non consente il corretto riconoscimento.

Figura 4.5 Immagini con errori di classificazione dovuti alla sogliatura

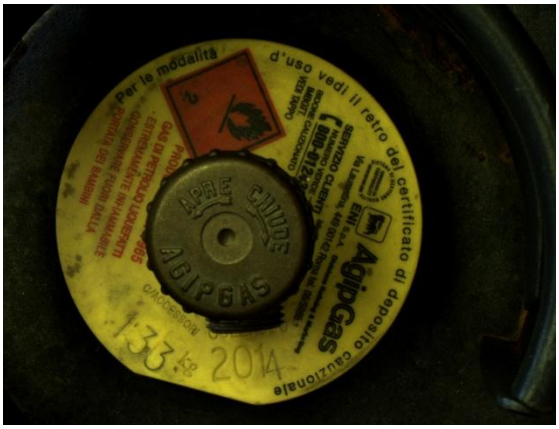
Analizzati i possibili problemi durante la classificazione è stata effettuata una analisi delle prestazioni del sistema esaminando ognuna delle tre sottofasi che costituiscono l'applicazione:

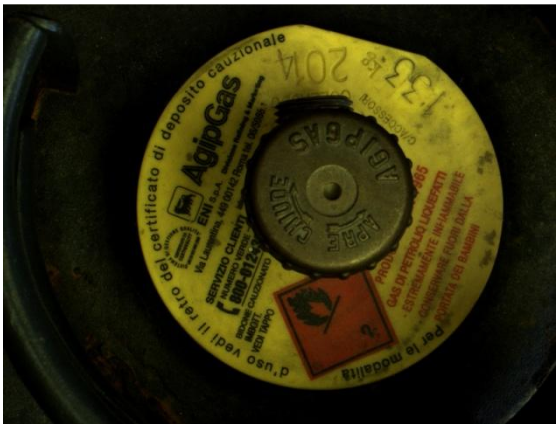
Algoritmo	**Riconoscimenti negativi**	**Errore di classificazione**
Localizzazione etichetta	1.10%	-
Individuazione data	3.56%	-
Riconoscimento data	-	9.25%

Come si può notare dalla tabella per le prime due fasi sono stati analizzati i riconoscimenti negativi, ovvero la percentuale dei casi in cui i due algoritmi non sono riusciti a raggiungere l'obiettivo per il quale sono stati preposti. La sottofase "Localizzazione etichetta" ha il compito di individuare l'intera etichetta, mentre "Individuazione data" ha l'incarico di identificare la data sfruttando la posizione del pittogramma. Nei risultati ottenuti rientrano proprio le immagini di scarsa qualità riportate nel Paragrafo 4.2.2. Per l'ultima delle tre sottofasi è stata utilizzata una colonna apposita in quanto si vuole stimare la percentuale dei casi in cui i numeri che costituiscono la data vengono assegnati alla classe sbagliata, ovvero le vere e proprie prestazioni del sistema. Il test è stato effettuato su circa trenta immagini. Il risultato ottenuto risulta essere abbastanza buono essendo l'applicazione sviluppata per fini ed in tempi didattici, ma essa è ancora lontana da un possibile utilizzo reale. Per raggiungere questo obiettivo è necessario ottimizzare le prestazioni del sistema, abbassando quindi l'errore di classificazione. Il fattore che può contribuire notevolmente a migliorare le prestazioni è la creazione di una fase di *tresholding* specifica per il tipo di immagini oggetto dell'elaborazione.

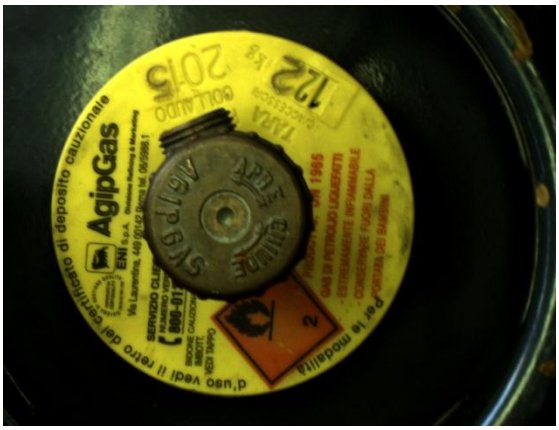
Di seguito vengono riportate alcune delle immagini utilizzate per il test:


Per le modalità
d'uso vedi il retro del certificato di deposito cauzionale
AgipGas
APRE CHIUDE AGIPGAS
TARA
PROSSIMO COLLAUDO
2015
12 kg


Per le modalità
d'uso vedi il retro del certificato di deposito cauzionale
AgipGas
APRE CHIUDE AGIPGAS
2014
13 kg


AgipGas
APRE CHIUDE AGIPGAS
2014
13 kg

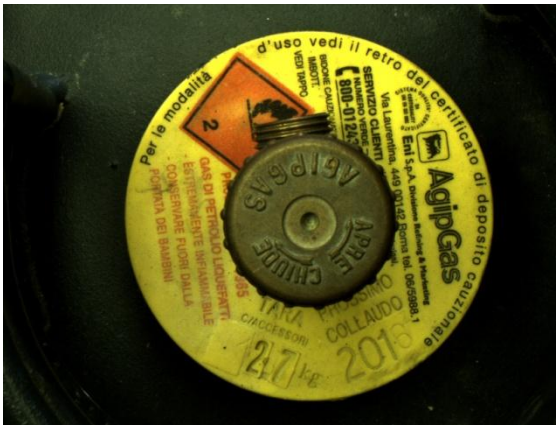

AgipGas
APRE CHIUDE AGIPGAS
2013
21 kg
GAS DI PETROLIO LIQUEFATTI
ESTREMAMENTE INFIAMMABILE
CONSERVARE FUORI DALLA PORTATA DEI BAMBINI


AgipGas
APRE CHIUDE AGIPGAS
2015
12 kg


AgipGas
APRE CHIUDE AGIPGAS
2013
12 kg


AgipGas
APRE CHIUDE AGIPGAS
2012
12 kg


AgipGas
APRE CHIUDE AGIPGAS
2012
12 kg


Per le modalità
d'uso vedi il retro del certificato di deposito cauzionale
AgipGas
APRE CHIUDE AGIPGAS
TARA C/ACCESSORI 24.7 kg
PROSSIMO COLLAUDO 2015


Per le modalità
d'uso vedi il retro del certificato di deposito cauzionale
AgipGas
AgipPetroli Business GPL
APRE CHIUDE AGIPGAS
TARA C/ACCESSORI 11.5 kg
PROSSIMO COLLAUDO 201


Per le modalità
d'uso vedi il retro del certificato di deposito cauzionale
AgipGas
APRE CHIUDE AGIPGAS
PROSSIMO COLLAUDO 2017


Per le modalità
d'uso vedi il retro del certificato di deposito cauzionale
AgipGas
APRE CHIUDE AGIPGAS
TARA C/ACCESSORI 20 kg
PROSSIMO COLLAUDO 2017


Per le modalità
d'uso vedi il retro del certificato di deposito cauzionale
AgipGas
APRE CHIUDE AGIPGAS
TARA C/ACCESSORI 13.9 kg
PROSSIMO COLLAUDO 2017


Per le modalità
d'uso vedi il retro del certificato di deposito cauzionale
AgipGas
APRE CHIUDE AGIPGAS
TARA C/ACCESSORI 12.1 kg
PROSSIMO COLLAUDO 2014


Per le modalità
d'uso vedi il retro del certificato di deposito cauzionale
AgipGas
APRE CHIUDE AGIPGAS
TARA C/ACCESSORI 13.4 kg
PROSSIMO COLLAUDO 201


Per le modalità
d'uso vedi il retro del certificato di deposito cauzionale
AgipGas
APRE CHIUDE AGIPGAS
TARA C/ACCESSORI 13.6 kg
PROSSIMO COLLAUDO 201

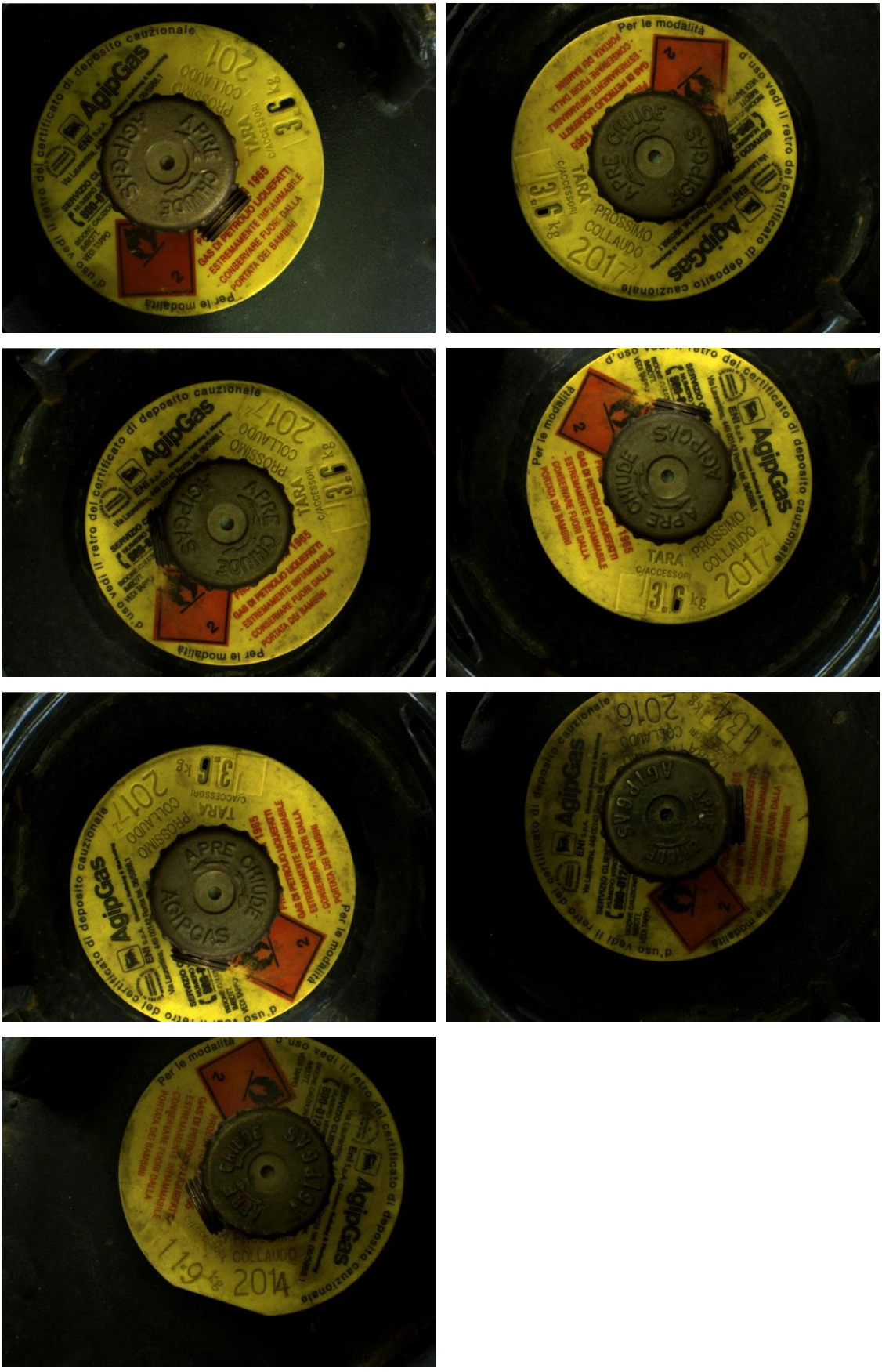


Figura 4.6 Immagini usate nei test

Bibliografia

[1] Duda R.O., Hart P., Stork D.G.: *Pattern Classification and Scene Analysis*. New York: John Wiley & Sons, 2000.

[2] Fu K.S.: *Syntactic Pattern Recognition and Applications*. Englewood Cliffs, N.J., Prentice-Hall, 1982.

[3] Jain A., Duin R., Mao J.: Statistical Pattern Recognition: A Review. *IEEE Transaction on Pattern Recognition and Machine Intelligence*, Vol. 22, n.1, 2000, p. 4-37.

[4] Pavlidis T.: Structural Pattern Recognition. NewYork, Springer-Verlag, 1977.

[5] Watanabe S.: Pattern Recognition: Human and Mechanical. New york: Wiley, 1985.

[6] Jain A.: *Fundamentals of Digital Image Processing*. Prentice-Hall.

[7] M. K. Hu.: Pattern recognition by moment invariants. In *Proc. IRE*, number 49, Sept. 1961.

[8] M. K. Hu.: Visual pattern recognition by moment invariants. *IEEE Comp. Society*, 1977.

[9] Gonzalez, R. C., and R. E. Woods: *Digital Image Processing*. Addison-Wesley, 1993.

[10] R. Jain, R. Kasturi, and B. G. Schunck: *Machine Vision*. McGraw-Hill, 1995.

[11] S. Impedove, L. Ottaviano, and S. Occhinegro: Optical character recognition – a survey. *Internal Journal of Pattern Recognition and Artificial Intelligence*, Vol. 5 (1-2), pp.1-24, 1991.

[12] E. A. Patrick: *Fundamentals of Pattern Recognition*. Prentice Hall, 1972.

[13] O. D. Trier, A. K. Jain, and T. Taxt: Feature extraction methods for character recognition – a survey. Pattern Recognition, Vol.29, No.4, pp 641-662, 1996.

[14] A. Khotanzad and Y. H. Hong: Invariant image recognition by Zernike moments, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.12, No.5, May 1990.

[15] D. Lazzaro: *Dispense di Tecniche di Elaborazione delle Immagini*, 2011/12.

[16] R. Cappelli: *Dispense di Visione Artificiale I*, 2011/12.

[17] D. Maltoni: *Dispense di Visione Artificiale II*, 2011/12.