Alma Mater Studiorum · Università di Bologna
Campus di Cesena

SCUOLA DI SCIENZE

Corso di Laurea in Ingegneria e Scienze informatiche

Progettazione e sviluppo di un sistema di rilevamento delle persone tramite l'uso di una videocamera termica

Relazione finale in:

PROGRAMMAZIONE DI RETI

Relatore:
Prof. GIOVANNI PAU

Presentata da:
SIMONE VENTURI

Seconda Sessione di Laurea
Anno Accademico: 2017-2018

Alla mia famiglia.

Indice

Introduzione

Lo scopo di questa tesi è quello di progettare ed implementare un sistema in grado di monitorare delle aree, percepire degli eventi, ad esempio la presenza di esseri umani, e registrarli opportunatamente in una base di dati, creando delle postazioni costituite da un *system on chip* provvisto di una videocamera comunicante con un server il quale svolge le funzioni più onerose dal punto di vista computazionale. Il software prodotto deve possedere una buona qualità e deve essere soprattutto estendibile, per possibili migliorie o aggiunte di funzionalità non previste in fase di analisi.
Nel primo capitolo vengono trattati i principali argomenti inerenti a questa tesi: una breve infarinatura riguardo il Machine Learning ed i suoi possibili utilizzi, in seguito un approfondimento sul tema centrale, ovvero la Computer Vision (o visione artificiale) in particolare la disciplina *Object Detection*.
Nel secondo capitolo espongo le principali tecnologie utilizzate nel progetto: la scheda che ho scelto, la videocamera termica ed il suo funzionamento, il principale linguaggio di programmazione adoperato, il relativo paradigma e la libreria.
Nel terzo capitolo analizzo e progetto il sistema in base ai requisiti ed alle funzionalità dello stesso partendo dalla postazione tipica in grado di poter interagire con il server e l'ambiente circostante. La progettazione del server riguarda, invece, l'accettazione, la gestione e l'elaborazione dei dati in entrata dalla postazione, rilevando la presenza umana nelle immagini ricevute e tenendone traccia sul database.
Nel quarto ed ultimo capitolo implemento il sistema seguendo quanto prodotto durante la fase precedente. Sarà eseguito un test su quanto implementato, rilevati i pregi ed i difetti, inoltre, sarà generato un nuovo classificatore a cascata per il rintracciamento di figure umane con la visione termica. Infine saranno paragonati i risultati dei due classificatori usando gli indici di Precisione e Recall.

Capitolo 1

Introduzione ai concetti

1.1 Machine Learning

Il Machine Learning (o apprendimento automatico) è una branca dell'intelligenza artificiale che ha come obiettivo quello di addestrare delle macchine affinché apprendano determinati algoritmi partendo da dei dati. L'idea principale, quindi, è quella di fornire un modello con molti esempi e da questi apprendere sviluppando un comportamento riproducibile su nuovi dati. Il set di dati fornito si divide in tre:

- il training set viene utilizzato per la fase di addestramento e contiene l'insieme di pattern su cui addestrare il sistema;
- il validation set è l'insieme dei pattern su cui tarare gli iperparametri;
- il test set è l'insieme di pattern su cui valutare le prestazioni finali del sistema.

L'apprendimento risulta essere Supervisionato se le classi dei pattern sono note a priori ed il set fornito per l'addestramento è etichettato, viceversa l'apprendimento può essere Non Supervisionato qualora le classi dei pattern non siano note e il set non etichettato, infine si parla di addestramento Semi-Supervisionato quando il training set è solamente in parte etichettato. L'apprendimento, inoltre, può essere raccolto in diverse categorie: si parla di Batch quando l'addestramento è eseguito una sola volta, dopo il quale il sistema non può più apprendere, è chiamato Incrementale quando dopo la prima sessione di apprendimento ne possono susseguire altre a rischio di dimenticare quanto appreso in precedenza, infine, si parla di apprendimento Naturale quando il sistema è in addestramento

continuo anche durante il periodo di attività, quest'ultima tipologia di apprendimento è analoga all'esperienza umana.

1.1.1 Machine Learning Pattern

I dati nel Machine Learning sono fondamentali perché il comportamento degli algoritmi non è predefinito ma appreso proprio da questi. I pattern sono suddivisi in:

- numerici: sono dei valori associati a caratteristiche misurabili, possono essere continui o discreti, oppure dei vettori;
- categorici: valori associati a caratteristiche qualitative e alla presenza/assenza di una caratteristica, non semanticamente mappabili in valori numerici;
- sequenze: pattern sequenziali con relazioni spaziali o temporali, come ad esempio un video (sequenza di frame);
- altri dati strutturati: output organizzati in strutture più complesse come alberi o grafi.

1.1.2 Problemi ricorrenti

Le tecniche di Machine Learning possono essere applicate per affrontare svariati problemi, tra cui:

- Classificazione: avviene quando vengono identificati schemi o insiemi di caratteristiche che definiscono il gruppo a cui appartiene un dato elemento. Il task di classificazione prende in input un gruppo di item ognuno già classificato e incluso in un insieme determinato e fornisce in output un modello (classificatore) che stabilisce l'appartenenza, basandosi sul valore che possiedono gli altri attributi;
- Regressione: specializzata nella previsione di variabili quantitative. Il concetto base su cui poggia l'intera teoria è la possibilità di approssimare la vera funzione che ha generato le osservazioni. La previsione è verificata sugli scarti d'errore generati dal modello rispetto ai veri valori;
- Clustering: detto anche analisi dei gruppi, consente di individuare gruppi (cluster) di pattern con caratteristiche simili non note a priori;

- Riduzione di dimensionalità: consiste nell'apprendimento di una mappatura da R^d a R^k, con $d > k$. La riduzione del numero di dimensioni dei pattern di input provoca una perdita di informazioni, questa perdita deve includere il più possibile le informazioni non importanti;

- Representation Learning: consiste nella rappresentazione dei pattern in termini di features, partendo da dei dati grezzi.

1.2 Visione artificiale

La visione artificiale (in inglese computer vision) è la disciplina che mira ad ottenere un modello tridimensionale approssimato del mondo reale partendo da immagini bidimensionali. La visione artificiale ha come scopo principale la riproduzione della vista umana intesa come metodo di interpretazione, comprensione ed elaborazione di informazioni contenute in una o più immagini. Questa disciplina si è sviluppata a partire dagli anni Sessanta in parallelo all'intelligenza artificiale, che studia più in generale i comportamenti automatici in grado di svolgere compiti autonomi, senza limitarsi ai processi di visione. Queste due discipline, che possono far sembrare una un caso particolare dell'altra, possiedono notevoli differenze metodologiche. Infatti, nell'intelligenza artificiale si presuppone l'esistenza di sistemi in grado di creare una rappresentazione simbolica del sistema reale e che questa possa essere utilizzata da algoritmi di manipolazione dati affinché venga prodotto un risultato, ovvero un comportamento da attuare. Nella visione artificiale, invece, la parte preponderante della ricerca è il metodo con il quale estrarre dal mondo reale la raffigurazione simbolica per produrre una conoscenza, a questo fine è necessario l'utilizzo dei due gradi di astrazione su cui si basa la visione artificiale: "early vision" e "high level vision". La prima, più consolidata, è il grado di astrazione più basso che è formata dagli algoritmi per estrarre dalle immagini le proprie caratteristiche come ad esempio l'estrazione dei bordi dagli oggetti oppure la convoluzione lineare, la seconda, invece, è il grado di astrazione più alto che si occupa di funzioni quali riconoscimento e categorizzazione.

1.2.1 Object Detection

Un classico problema della visione artificiale è il rilevamento di determinati oggetti in una scena con un altro grado di affidabilità. I metodi grazie ai quali è possibile conseguire questo obiettivo è tramite l'uso di machine learning oppure con un approccio deep learning; il concetto alla base sul quale puntare è che ogni oggetto ha determinate caratteristiche grazie alle quali è possibile classificarlo.
Mi concentrerò su un approccio machine learning, in particolare il framework proposto da Paul Viola e Michael Jones basato su Haar features. I due ricercatori hanno proposto un algoritmo molto robusto ed utilizzabile in applicazioni real time che comprende quattro fasi:

- selezione delle haar features;
- creazione di un'immagine integrale;
- addestramento adaptive boosting (AdaBoost);
- classificatori a cascata.

Le caratteristiche ricercate saranno presenti sull'immagine sotto forma di somme relative ad aree di rettangoli. Come mostra la Figura 1.1, il valore della *feature* è ottenuto sottraendo la somma dei valori nell'area più scura da quelli contenuti nella zona più chiara.

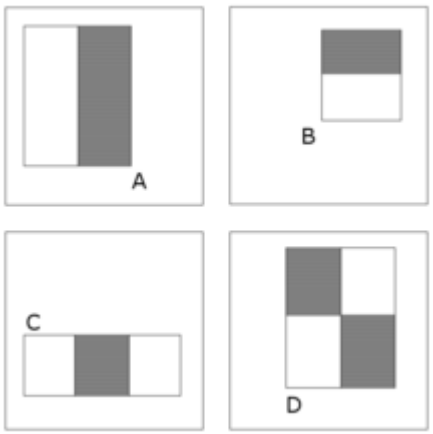

Figura 1.1: Principali pattern di ricerca

Per trovare le caratteristiche, quindi, è necessario calcolare continuamente ogni possibile zona grigia e sottrarle la relativa zona bianca a seconda del pattern che si vuole utilizzare. Questo processo renderebbe l'algoritmo inutilizzabile a causa dell'alto costo computazionale, perciò i due ricercatori ebbero la brillante idea di utilizzare una struttura dati chiamata *summed-area table* che chiamarono immagine integrale, ovvero una lookup table bidimensionale delle stesse dimensioni della matrice di partenza.
Sia I una generica immagine in scala di grigi. Il valore del pixel (x, y) dell'immagine integrale $\mathcal{I}$ rappresenta la somma dei valori di ogni pixel dell'immagine sorgente contenuti all'interno del rettangolo $(0, 0) - (x, y)$:

$$\mathcal{I}(x, y) = \sum_{v=0}^{y} \sum_{u=0}^{x} I(u, v)$$

Perciò ogni rettangolo ora è calcolabile come una somma di quattro elementi:

$$\sum_{y=y_0}^{y_1} \sum_{x=x_0}^{x_1} I(x, y) = \mathcal{I}(x_1, y_1) + \mathcal{I}(x_0 - 1, y_0 - 1) - \mathcal{I}(x_1, y_0 - 1) - \mathcal{I}(x_0 - 1, y_1)$$

Nonostante questa riduzione del carico, resta comunque proibitivo poter valutare ogni caratteristica possibile su ogni immagine per cui viene utilizzata una tecnica di *boosting* che consiste nel dare un peso in base all'importanza rilevata di una determinata caratteristica, costituendo così dei classificatori forti e deboli. Un ultimo punto di forza di questo framework è l'architettura dello stesso, ovvero a cascata. I classificatori vengono raggruppati in diverse fasi, nelle prime ci saranno quelli più facili da calcolare per velocizzare gli scarti delle sotto-aree negative ed utilizzare il tempo e le risorse solo per quelle di interesse, successivamente ci saranno sempre più classificatori. Per essere riconosciuto un oggetto deve superare ogni fase, senza mai fallire per questo motivo il metodo fornito dai due ricercatori è utilizzabile in real time e genera pochi falsi positivi.
A tal proposito, due indici utili per determinare la qualità di un classificatore sono:

- **Precisione:** la percentuale che determina la veridicità di previsione positiva, calcolata come

$$PositiviReali/(PositiviReali + FalsiPositivi)$$

- **Recall:** la percentuale che indica la frequenza con la quale un oggetto venga effettivamente rilevato, calcolata come

$$PositiviReali/(PositiviReali + FalsiNegativi)$$

Capitolo 2

Tecnologie usate

In questo breve capitolo introdurremo la tecnologia necessaria al conseguimento dell'obiettivo. Nelle prime sezioni mostrerò l'hardware utilizzato, in seguito il linguaggio prevalentemente usato, il paradigma ad oggetti e la libreria adottata.

2.1 Raspberry Pi 3

Il Raspberry Pi 3 è un single board computer, ovvero una scheda che ospita un calcolatore, molto diffuso per uso scolastico grazie alla sua versatilità e al suo costo contenuto. A livello Hardware, il sistema su circuito integrato (SoC) Broadcom BCM2837 incorpora una CPU ARM Cortex-A53 quad-core con una frequeza di 1.2 GHz a 64 bit, 1 GB di SDRAM condivisa con il chip grafico, una porta Ethernet 10/100 Mb/s, WLAN 802.11n, Bluetooth 4.1; a livello software, invece, la scheda è progettata per ospitare sistemi operativi basati sul kernel Linux, come ad esempio Raspbian.

2.2 Videocamera termica Flir Lepton 3.5

La videocamera termica Flir Lepton 3.5, in Figura 2.1, è una videocamera che integra un gruppo di lenti a fuoco fisso con un sensore di matrice 160x120, data la sua dimensione molto ridotta (quanto una moneta), il basso consumo energetico e il costo accessibile, risulta un ottimo prodotto per la funzione che deve svolgere. La videocamera dispone di otto pin:

- due per l'alimentazione e il collegamento a terra (3.5V e GND);

- quattro per la comunicazione SPI: il selettore dello slave (CS), il clock (CLK) e i due canali per la comunicazione unidirezionale (MOSI e MISO);
- due per la comunicazione I2C: il canale dei dati (SDA) e quello del clock (SCL).

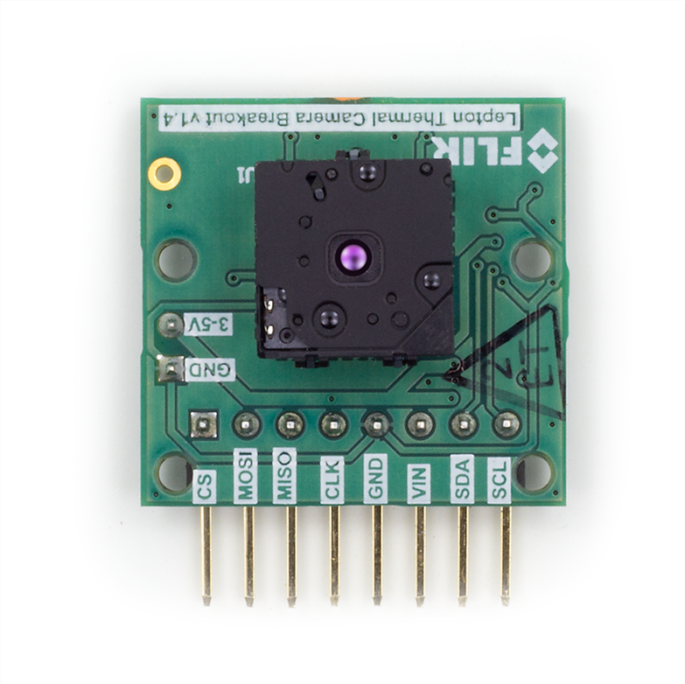

Figura 2.1: videocamera termica Flir Lepton 3.5

2.2.1 Funzionamento videocamera termica

Come mostra la figura 2.2, il gruppo ottico focalizza la radiazione infrarossa dalla scena su una serie di rivelatori termici di 17 micrometri, ognuno di questi è un micro-bolometro di ossido di vanadio (VOx) la cui temperatura varia in risposta al flusso incidente. Il cambiamento di temperatura provoca una variazione proporzionale alla resistenza posta dal bolometro, inoltre l'ossido di vanadio fornisce un alto coefficiente di resistenza alla temperatura (TCR), mostrando, così, una eccellente termo-sensibilità. L'array di micro-bolometri è posto sopra un circuito di lettura integrato che comprende l'intero piano focale, ottenendo così una matrice di valori a 14-bit pronta per essere trasmessa tramite la comunicazione SPI alla scheda alla quale è collegata la videocamera.

L'interfaccia CCI (simile a I2C) è utilizzata come canale per la comunicazione di controllo e di impostazione della videocamera.

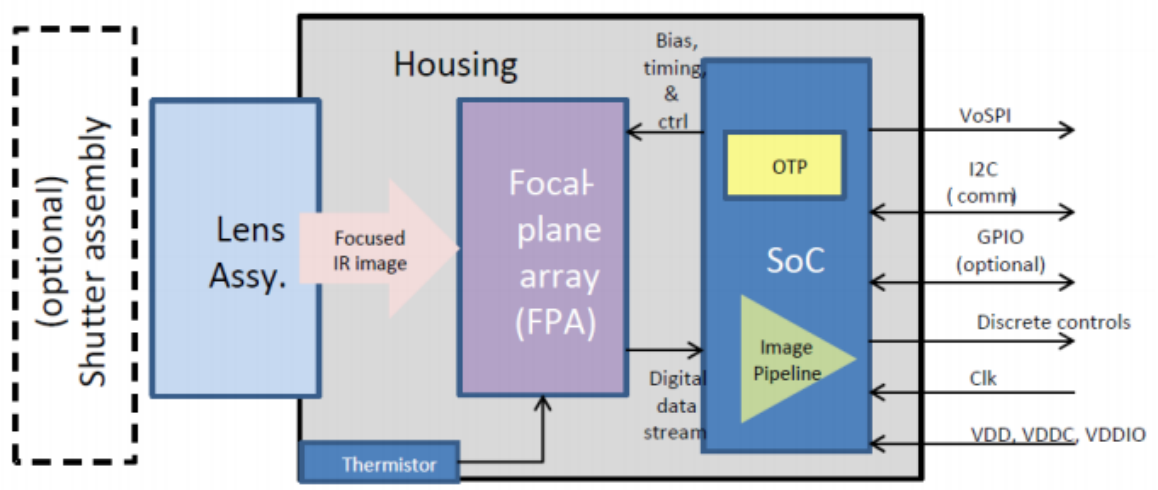

Figura 2.2: Schema funzionamento videocamera Flir Lepton 3.5

2.3 Java e OOP

Java è un linguaggio di programmazione ad alto livello, tipizzato e orientato agli oggetti, progettato specificatamente per essere indipendente il più possibile dalla piattaforma sulla quale è eseguito grazie all'uso di un interprete, la Java Virtual Machine. Secondo il proprio sito internet Java venne creato per soddisfare cinque obiettivi primari:

- essere "semplice, orientato agli oggetti e familiare;
- essere "robusto e sicuro";
- essere indipendente dalla piattaforma ("Write Once, Run Anywhere");
- contenere strumenti e librerie per il networking;
- essere progettato per eseguire codice da sorgenti remote in modo sicuro.

Senz'ombra di dubbio la caratteristica fondamentale di Java è l'indipendenza dalla piattaforma di esecuzione, l'uso della JVM è una soluzione che migliora la portabilità del codice, in quanto la compilazione del codice produce un bytecode (un linguaggio intermedio più astratto rispetto il linguaggio macchina) eseguibile su ogni JVM e genera dei codici oggetto specifici per ogni sistema operativo, ma al tempo stesso rende meno efficiente l'uso delle risorse poiché c'è un livello intermedio aggiuntivo. Altri vantaggi che si possono ottenere usando Java come linguaggio di programmazione, oltre quelli già citati, sono:

- abbondanza di librerie;
- compatibilità con sistemi palmari ed embedded;
- alta integrazione con il web;
- peculiarità del paradigma ad oggetti

A tal proposito, il paradigma ad oggetti rende possibile una modellazione del dominio di interesse e fornisce i mezzi per creare un codice di qualità soprattutto in termini di riuso. I principi su cui si fonda la programmazione ad oggetti sono:

- Classe: è la rappresentazione di un insieme di oggetti "simili", caratterizzata da una struttura per i dati e da un'interfaccia che definisce quali sono le operazioni associate agli oggetti, ovvero l'insieme dei servizi implementati. Tutti gli oggetti di una classe hanno gli stessi attributi e metodi. Esistono metodi di due tipi: quelli che restituiscono astrazioni significative sullo stato dell'oggetto cui sono applicati, e quelli che ne alterano lo stato;
- Oggetto: è l'istanza di una classe, sono gli elementi di base del paradigma, e corrispondono ad una entità del dominio applicativo. A sua volta questo è costituito da:
  - una identità (OID, Object Identifier) che gli viene assegnata alla creazione ed è indipendente dallo stato attuale ed è, inoltre, non alterabile;
  - uno stato definito come l'insieme dei valori assunti dai propri attributi in un determinato lasso temporale;
  - un comportamento definito come l'insieme delle operazioni.

- Ereditarietà: è il meccanismo fondamentale del paradigma ad oggetti in quanto permette ad una classe di derivare da una o più classi caratteristiche già definite, alterarle all'occorrenza e creare così un nuovo tipo di dato. Questo meccanismo è solamente in parte implementato in Java poiché non è prevista l'ereditarietà multipla, al contrario di altri linguaggi come ad esempio C++;

- Polimorfismo: è un meccanismo che permette di creare metodi con lo stesso nome ma con implementazioni differenti, questo è reso possibile solo attraverso l'overload che produce metodi omonimi ma con signature differenti. Un'altra peculiarità del polimorfismo è la possibilità di ridefinire un metodo di una sottoclasse attraverso l'override.

- Istanziamento dinamico (o late binding): l'abbinamento dell'istanziamento dinamico con il polimorfismo permette che un oggetto abbia un determinato comportamento a seconda della classe da cui deriva, questo è un enorme passo avanti poiché questo avviene a tempo di esecuzione e non si è più vincolati ad una determinata implementazione a tempo di compilazione come avviene ad esempio nel linguaggio C.

- Delegazione: avviene quando un oggetto complesso contiene al suo interno un riferimento ad un altro oggetto, per cui l'oggetto complesso delegherà alcune funzioni all'oggetto contenuto. Questo meccanismo è fondamentale per l'espressione delle associazioni tra oggetti.

2.4 Libreria OpenCV

OpenCV (acronimo di Open Source Computer Vision Library) è una libreria software multipiattaforma orientata principalmente alla visione artificiale in tempo reale e all'apprendimento automatico. Distribuita sotto licenza BSD, inizialmente fu sviluppata da Intel come strumento di testing per le nuove CPU, una sua primissima versione alfa fu rilasciata nel 2000 mentre la versione 1.0 uscì nel 2006. Gli obiettivi principali del progetto furono:

- fornire un codice open source ed anche ottimizzato per il riconoscimento;

- diffondere la conoscenza della visione fornendo un'infrastruttura comune, sulla quale gli sviluppatori possano poi svilupparne altra, trasferibile e comprensibile;
- creare applicazioni commerciali basate sulla visione con una licenza che non richieda che il codice sia libero.

La libreria è scritta con il linguaggio di programmazione C++, ma è possibile interfacciarsi anche attraverso C, Python e Java.

Capitolo 3

Analisi e progettazione del sistema

L'obiettivo del progetto è quello di realizzare un sistema da esterno con il quale monitorare delle zone, rilevare presenze umane e tenerne traccia. La soluzione che ho ritenuto migliore è quella di creare delle piccole postazioni costituite da una scheda Raspberry Pi alla quale è collegata una videocamera termica, questa scelta risulta vantaggiosa poiché, una volta installata, ogni postazione può essere controllata da remoto e non risulta necessario smontarla oppure accedervi fisicamente, inoltre la postazione, così impostata, è operativa in ogni condizione atmosferica, sia di giorno che di notte e richiede un consumo energetico contenuto, soprattutto se le risorse saranno utilizzate efficientemente. Il server, invece, sarà in ascolto pronto ad accettare le richieste in arrivo, processare le immagini ricevute, determinare la presenza o meno di un individuo, registrare la presenza sul database. Un aspetto non trascurabile di questa implementazione è senz'altro la sicurezza, infatti, accedendo alla rete sulla quale sono connessi le postazioni ed il server, è possibile danneggiare il sistema o appropriarsi di informazioni e dati impropriamente.

3.1 Progettazione della postazione

La postazione presenta la videocamera collegata al Raspberry sul quale dovrà essere implementato un software in grado di ottenere l'immagine dalla termocamera, salvarla ed inviarla al server. Il comportamento specificato sopra può essere rappresentato facilmente tramite macchina a stati finiti sincrona in Figura 3.1 prodotta durante la fase di progettazione, i tre stati contengono le tre macro-operazioni da svolgere e, una volta concluse, lasciano il controllo allo stato successivo.

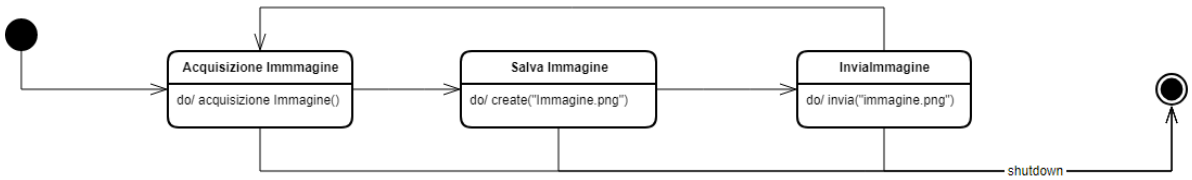

Figura 3.1: Macchina a stati finiti della postazione

3.2 Progettazione del server

Il server deve accettare le richieste multiple che possono arrivare dalle varie postazioni perciò è necessario implementare una socket di benvenuto che resti in attesa di essere contattata per poi delegare la trasmissione dei dati ad una socket ausiliaria, questo è necessario perché il server deve essere reattivo e deve rispondere ad ogni richiesta, inoltre, per avere un canale affidabile sul quale trasmettere, ho scelto di usare il protocollo di trasporto TCP perché mi garantisce la trasmissione di ogni pacchetto senza che si verifichino errori o mancate ricezioni. In secondo luogo, il server dopo aver salvato sul filesystem l'immagine ricevuta deve riuscire a leggerla e ad elaborarla affinché riesca a capire se vi è o meno la presenza di un individuo, a questo proposito tornerà molto utile la libreria di visione artificiale OpenCV. Infine, dopo aver determinato o meno una presenza umana, il server deve registrare sul proprio db l'evento.

3.2.1 Approfondimento sulla progettazione della base di dati

In questa breve sezione, spiegherò per quale motivo risulta fondamentale l'implementazione di un piccolo database per il conseguimento del mio obiettivo. La base di dati aiuta a conservare e tenere traccia dei dati, gestire automaticamente gli accessi e fornire uno strumento di sicurezza ulteriore, essendo protetto da password. Si denota, quindi, una soluzione di gran lunga migliore rispetto, ad esempio, ad un semplice file di registrazione. Per i motivi sopraelencati, l'implementazione del database migliorerebbe l'applicazione dal punto di vista della robustezza perché a fronte di possibili accessi multipli garantirebbe un comportamento corretto e deterministico. Inoltre, dal punto di vista dell'estendibilità, il database, renderebbe possibili le implementazioni di altre

funzionalità, come ad esempio il controllo remoto del sistema da un'applicazione per smartphone.

3.2.2 Progettazione concettuale del database

Come mostra la Figura 3.2, la videocamera, se attiva, rileva continuamente immagini, questa è posta in una determinata postazione che controlla un'area determinata, qualora nell'elaborazione ci fosse qualcosa di cui tenere traccia, viene generato un evento che possono essere raggruppati in diverse tipologie.

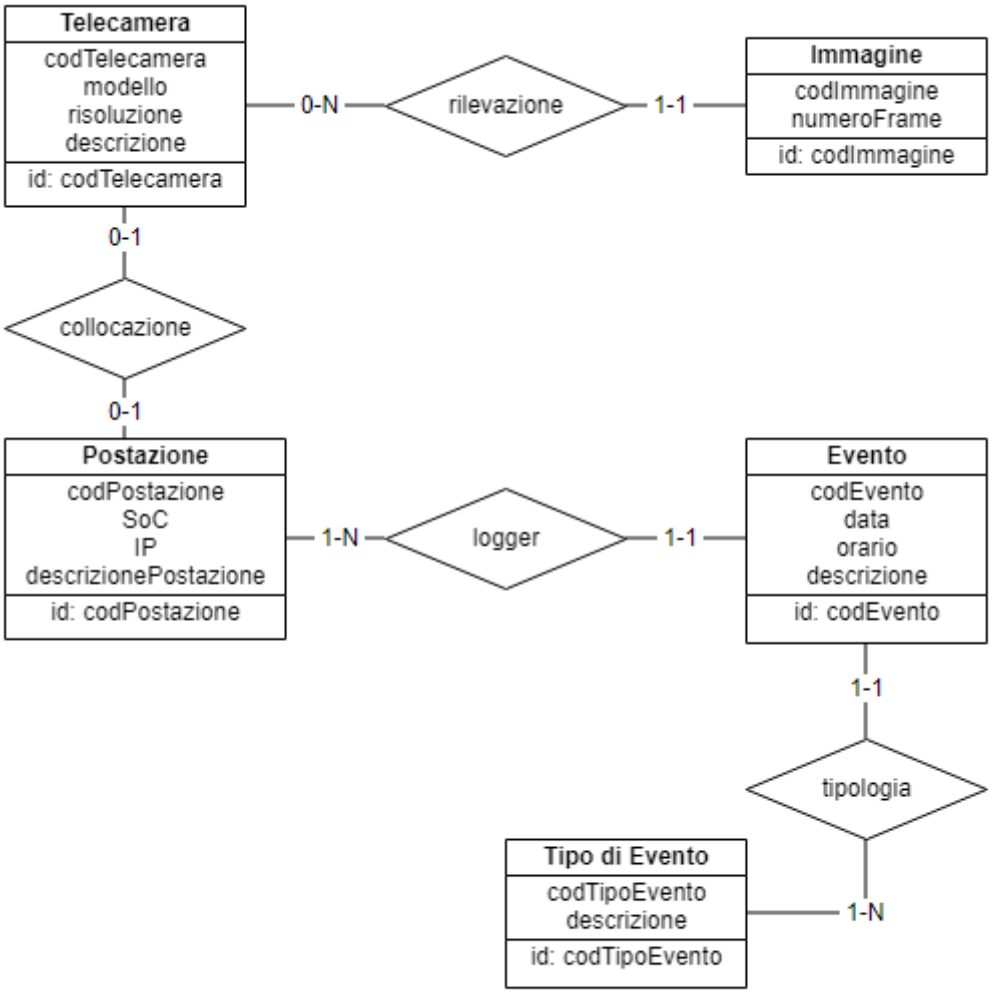

Figura 3.2: Diagramma Entity-Relationship della base di dati

3.2.3 Progettazione logica del database

La progettazione logica del database facilita di gran lunga l'implementazione finale dell'applicativo, una corretta risoluzione delle relazioni trovate in fase di progettazione concettuale migliora, inoltre, la qualità del software. Durante questa fase ho prodotto il seguente schema logico:

Evento (codEvento, data, orario, descrizione, codTipoEvento: TIPO DI EVENTO, codPostazione: POSTAZIONE)

Immagine (codImmagine, numeroFrame, codTelecamera: TELECAMERA)

Postazione (codPostazione, SoC, IP, descrizionePostazione, codTelecamera: TELECAMERA)

Telecamera (codTelecamera, modello, risoluzione, descrizione, codPostazione: POSTAZIONE)

Tipo di Evento (codTipoEvento, descrizione)

Non essendoci particolari soluzioni di relazioni degne di nota, vorrei soffermarmi sulla ridondanza che ho preferito mantenere. Nella relazione "collocazione", in Figura 3.2, ho scelto di inserire le chiavi importate in entrambe le entità perché le istanze di queste due classi non sono numerose, al contrario degli accessi che possono essere risparmiati in determinate ricerche.

Capitolo 4

Implementazione del sistema

In questa sezione verrà esposta l'implementazione del prototipo seguendo la progettazione ideata nel capitolo precedente.

4.1 Sviluppo della postazione

La postazione deve eseguire le stesse operazioni continuamente, acquisire l'immagine dalla videocamera tramite la comunicazione SPI, salvarla in locale, inviarla al server. Constatando che ogni immagine occupa all'incirca 10KB ho deciso di mantenere al massimo 100 frame come una sorta di backup locale, mentre per quanto riguarda l'invio ho scelto un buffer di 1KB. Come mostra il codice riportato qui sotto, aggiungendo alla libreria pylepton le specifiche descritte sopra si può ottenere una postazione creata per la videocamera termica FLIR Lepton 3.5.

```
#!/usr/bin/env python

import sys
import numpy as np
import cv2
import os
from pylepton.Lepton3 import Lepton3
import socket
```

```
def capture(flip_v = False, device = "/dev/spidev0.0"):
  with Lepton3(device) as l:
    a,_ = l.capture()
  if flip_v:
    cv2.flip(a,0,a)
  cv2.normalize(a, a, 0, 65535, cv2.NORM_MINMAX)
  np.right_shift(a, 8, a)
  return np.uint8(a)

  counter=0
  path = '/home/pi/Desktop/pylepton-lepton3/img'
  while 1:
    image = capture(flip_v = options.flip_v,
                    device = options.device)
    cv2.imwrite(os.path.join(path ,
                'img'+ str(counter%100) + '.png'), image)
    s = socket.socket()
    s.connect(("192.168.0.101",9001))
    f = open (path + "/img"  + str(counter%100) + ".png", "rb")
    fi = f.read(1024)
    while (fi):
        s.send(fi)
        fi = f.read(1024)
    s.close()
    counter = counter +1
```

4.2 Sviluppo del server

Il server ha un comportamento più articolato rispetto alla postazione, deve essere in ascolto costantemente sulla porta scelta, accettare le connessioni, scartare quelle non riconosciute dal proprio database, salvare le immagini in arrivo, elaborarle e registrare gli eventi rilevati in queste ultime. La socket TCP resta in ascolto sulla porta 9001, attende

di essere contattata da una postazione e non appena riceve un segnale dal client lancia un nuovo thread e delega il compito ad un'altra socket con il metodo mostrato nella Figura 4.2. Il server prima di accettare il trasferimento del file si assicura che l'indirizzo dal quale ha ricevuto la richiesta corrisponda all'indirizzo di una postazione, successivamente, avendo avuto un riscontro positivo prosegue con la ricezione ed il salvataggio dell'immagine.

```
public void run() {
    /*controllo che il client appartenga ad una postazione*/
    String clientIp = socket.getRemoteSocketAddress().toString();
    List<Postazione> postazioni = new PostazioneTable().findAll();
    boolean found = false;
    for(Postazione p : postazioni){
        if (p.getiP().equals(clientIp)){
            found = true;
        }
    }
    /*salvo il contenuto solo se la postazione è stata riconosciuta*/
    if(found){
        try {
            InputStream inputStream = socket.getInputStream();
            File saveFile = new File(PATH + "\\" + counter + ".png");
            FileOutputStream fos = new FileOutputStream(saveFile);
            int i = 0;
            while((i=inputStream.read(buf))!=-1) {
                  fos.write(buf, 0, i);
            }
            fos.close();
            CounterUtilities.checkImage(PATH, counter);
        } catch (Exception e) {
            e.printStackTrace();
        }   finally {
            try {
                socket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```

Figura 4.1: Implementazione metodo run()

4.2.1 Implementazione del database

L'applicazione, scritta in Java, ha un'architettura semplice ma che allo stesso tempo mantiene una netta distinzione fra la parte che gestisce la ricezione dei dati e la parte relativa alla gestione e alla mappatura del database. Ogni entità e relazione della base di dati viene mappata nell'applicazione tramite una propria classe java dove si trovano i concetti di attributi e chiavi tradotti come tipi di dato (string, int, long ecc.). Ogni tabella viene a sua volta esplicitata tramite una classe la quale comunica con le classi che mantengono gli aspetti relativi alle entità che partecipano alla tabella stessa. All'interno delle classi per le tabelle, vi sono tutti i metodi necessari per gestire le operazioni di sistema (fra i più comuni: creazione della tabella, ricerca ti tutti gli elementi, ricerca per chiave), i quali sono formati da query MySql per la comunicazione con il database. Le query sono state scritte in modo tale da semplificare le operazioni in Java e la gestione delle operazioni a livello di applicazione. Per esempio nelle ricerche vengono salvati tutti gli elementi di una tupla in modo tale da costruire un oggetto Java completo sul quale è possibile effettuare molte operazioni con estrema facilità.

4.2.2 Rilevamento delle persone

Il rilevamento è affidato al metodo *checkImage(String path, int nFrame)*, i due parametri di input consentono alla funzione di ritrovare univocamente il frame appena ricevuto. Dopo aver letto l'immagine dal filesystem, questa viene ispezionata con l'aiuto della libreria OpenCV, più precisamente con il metodo *detectMultiScale(Mat image, MatOfRect objects, double scaleFactor)*, con l'uso della classe CascadeClassifier e con un classificatore, se l'ispezione rileva una o più persone, queste vengono inserite nella collezione objects. Il classificatore usato dalla libreria OpenCV è un Haar-like feature: si basa sul concetto di analisi di un immagine tramite la suddivisione di quest'ultima si sezioni vicine di dimensioni rettangolari calcolando la somma dell'intensità dei pixel e successivamente applicare una sottrazione tra le somme, cercando così di trovare una logica nella differenze di intensità delle diverse zone adiacenti. Il classificatore che ho utilizzato in questa fase è quello fornito dalla libreria stessa, è stato addestrato su immagini scelte da loro, perciò non è garantito che abbia un comportamento ottimo su immagini con rilevazione termica poiché hanno pattern diversi. Una volta avvenuta la rilevazione ho ritenuto degni di nota due tipologie di eventi da tracciare:

- la ricezione di un file non leggibile, utile ad esempio a rilevare malfunzionamenti in una determinata postazione;

- il riconoscimento di un numero di individui diverso dal frame precedente. Questa scelta è dovuta al metodo utilizzato per recuperare frame, infatti utilizzando una macchina a stati finiti sincrona nella postazione, viene effettuato un polling a 8,7 Hz, perciò con la stessa frequenza verranno fatte le valutazioni nel server risultando così poco sensato tenere traccia costantemente dello stesso numero di individui nei successivi frame.

4.3 Test dell'applicazione

Dopo aver ottenuto un prototipo dell'applicazione, ho proceduto con una fase di test durante la quale ho implementato una piccola interfaccia grafica dalla quale potessi avere un feedback istantaneo e valutare personalmente il lavoro fin qui svolto. Una volta lanciato il server, ho acceso anche la postazione che ha iniziato ad acquisire ed inviare i frame, i quali arrivano correttamente e vengono salvati in un percorso specifico del filesystem in base all'indirizzo IP della postazione. Al rilevamento di un essere umano o alla ricezione errata di un'immagine, l'inserimento di nuovi record nella base di dati è eseguito correttamente.

4.3.1 Risultati ottenuti

L'applicazione creata ha un comportamento abbastanza robusto: durante la fase di test non ha mai avuto problemi ed ha proseguito la propria esecuzione fino alla mia interruzione, inoltre la postazione scarta automaticamente la maggior parte dei frame acquisiti non idonei (quelli con l'otturatore semi-chiuso o quelli causati da errori a livello hardware) e quei pochi che vengono trasmessi con errori vengono evidenziati opportunamente e registrati nel database. Il sistema gestisce egregiamente l'uso delle risorse soprattutto grazie anche alla distribuzione del calcolo più pesante alla parte server, inoltre la produzione del codice in moduli distinti sin dalla fase di progettazione ha reso il software decisamente riusabile ed estendibile per possibili usi futuri. Inoltre, un ulteriore punto di forza dell'applicazione creata è nella portabilità della stessa: l'uso di java nella parte server permette a questa di poter essere lanciata su qualsiasi macchina in possesso

di un ambiente java, anche se va ad incidere sull'efficienza; la postazione invece è creata ad hoc per la tipologia di hardware in possesso.

4.3.2 Problemi rilevati

Il software prodotto pur avendo un comportamento corretto ed aderente alle specifiche proposte non è esente da problemi. Il principale difetto è legato all'affidabilità: questa applicazione nasce con il presupposto di poter essere utilizzata per il rintracciamento di essere umani e potrebbe essere adoperata in sistemi contro le intrusioni, ad esempio per la sorveglianza domestica, perciò, date queste premesse, è indispensabile che questa riesca a riconoscere una figura umana con una buona percentuale di certezza. Durante la fase di test ho potuto constatare che l'applicazione avesse il grande difetto di produrre dei falsi negativi, come mostra la Figura 4.2, ovvero dei frame nei quali ci fosse una figura umana ma che il software non fosse in grado di comprendere che questa lo sia.

Figura 4.2: Mancate rilevazioni del sistema

4.4 Creazione di un classificatore a cascata

Valutando i risultati ottenuti, notando che i classificatori usati in precedenza non fossero sufficientemente efficaci, ho ritenuto opportuno creare un classificatore ad hoc per immagini termiche con l'obiettivo di riconoscere una persona a mezzobusto. Grazie all'uso della libreria OpenCV, è possibile addestrare un classificatore personalizzato per ogni oggetto a patto che si abbia a propria disposizione un set di immagini abbastanza vasto, alcune con l'elemento di nostro interesse che chiameremo "positive", altre senza che saranno le "negative". Innanzitutto ho selezionato una trentina di immagini

termiche grazie all'ausilio della postazione nelle quali ci fosse almeno una figura umana a mezzobusto, successivamente ho cercato in rete un set di 400 immagini in scala di grigi nelle quali non comparisse nessuna figura umana. Ho creato un file di testo all'interno del quale ho specificato per ogni immagine il numero di oggetti di interesse contenuti e le relative coordinate all'interno della figura stessa, dopodiché grazie al comando *opencv_createsamples* ho prodotto per ogni immagine positiva sessanta nuove figure nelle quali ci fosse come sfondo un'immagine negativa e all'interno, in una posizione casuale, l'elemento precedentemente segnato con una variazione del proprio angolo e della propria luminosità originale come mostra la Figura 4.3. Così facendo ho ottenuto più di 1800 immagini positive.

Figura 4.3: Creazione di due immagini positive

Con lo stesso comando ma con parametri diversi ho costruito un file vettore dentro il quale ci fossero solamente le immagini ritagliate in maniera che risaltassero gli elementi dai quali estrapolare le caratteristiche come mostrato nella Figura 4.4, in questo caso la sagoma a mezzobusto.

Figura 4.4: Esempio di vettore

Giunti a questo punto, tramite il comando *opencv_traincascade* con dieci fasi specificate ho ottenuto il mio nuovo classificatore.

4.5 Confronto dei risultati

Per confrontare il funzionamento dell'applicazione con i due diversi classificatori ho scelto di utilizzare due sequenze ognuna con 400 frame, contenenti momenti in cui non ci saranno elementi da rilevare ed altri con persone. La differenza sostanziale tra le due sequenze è l'ambiente circostante: in una è percepita una temperatura di 13° C, nell'altra di 22° C.
Come mostra la Tabella 4.1, il classificatore fornito dalla libreria durante il test in ambiente più fresco aveva grossi problemi legati all'identificazione di figure umane infatti su 291 frame con persone solamente 42 sono stati rilevati correttamente, invece come nota positiva non ha mai prodotto rilevazioni su oggetti sbagliati. Il classificatore addestrato personalmente ha prodotto risultati migliori sulla stessa sequenza infatti sulle 291 immagini 209 sono state rilevate correttamente, senza produrre ancora nessun falso positivo.
Si può facilmente notare che la precisione, ovvero la percentuale con la quale il sistema predice un risultato effettivamente positivo sia del 100% in entrambi i classificatori, mentre l'indice di recall, cioè la percentuale dei rilevamenti rispetto a tutti quelli possibili sia differente:

- con il classificatore fornito $Recall_f = 42/291 = 14{,}4\%$;
- con il classificatore addestrato $Recall_c = 209/291 = 71{,}8\%$.

Tabella 4.1: Risultati con ambiente a 13° C

	Vero positivo	Falso negativo	Vero negativo	Falso positivo
Classificatore fornito	42	249	109	0
Classificatore creato	209	82	109	0

L'altra serie di immagini ha prodotto risultati in linea con quelli precedenti, come mostra la Tabella 4.2 su 400 frame, 267 contengono presenze umane ma solamente in 28 di questi vengono rilevati dal classificatore di OpenCV e 172 del classificatore creato ad hoc. Nuovamente nessun frame è stato rilevato come positivo senza contenere sagome umane.
Mentre la precisione rimane identica al caso precedente l'indice di Recall risulta in entrambi i casi inferiore:

- $Recall_f = 28/267 = 10{,}5\%$;
- $Recall_c = 172/267 = 64{,}4\%$;

Tabella 4.2: Risultati con ambiente a 22° C

	Vero positivo	Falso negativo	Vero negativo	Falso positivo
Classificatore fornito	28	239	133	0
Classificatore creato	172	95	133	0

Le variazioni dei risultati relative ai due test sono riconducibili sia alla differenza dei movimenti e delle scene contenute, sia al minore contrasto evidenziato dalle immagini, infatti la Figura 4.5 mostra come siano diverse due immagini con una decina di gradi di temperatura di differenza.

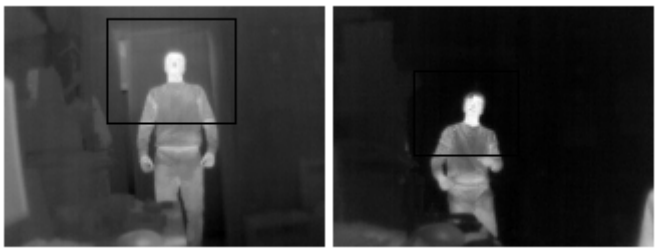

Figura 4.5: Variazione della temperatura dell'ambiente circostante

Conclusione

L'obiettivo di questa tesi era quella di progettare ed implementare un sistema in grado di percepire la presenza umana nelle zone di interesse con una buona percentuale di attendibilità. Il progetto creato è aderente alle specifiche, ha un comportamento robusto e, in seguito all'addestramento del nuovo classificatore, è anche sufficientemente affidabile, poiché considerando che una postazione produce almeno 8 immagini al secondo se in questo lasso di tempo fosse presente una persona la probabilità che questa non venga rintracciata è vicinissima allo 0%. Infatti secondo la distribuzione geometrica la probabilità che il primo successo, in questo caso la rilevazione, avvenga con l'esecuzione di 8 prove indipendenti, ognuna con la stessa probabilità di successo (0,718) è:

$$P(X = 8) = 0,718(1 - 0,718)^{8-1} \simeq 0,01\%$$

Il software presentato non è comunque privo di migliorie e di possibili estensioni: una tra tutte l'interfacciamento tramite applicazione per smartphone che si presta molto bene per un uso legato alla sicurezza domestica, con una implementazione di notifiche generate da specifici eventi registrati. Un altro possibile miglioramento potrebbe consistere nell'addestramento di un nuovo classificatore che offra prestazioni migliori di quello creato, soprattutto in condizioni diverse da quelle proposte.

Ringraziamenti

Vorrei ringraziare il professore Giovanni Pau per avermi dato la possibilità di sviluppare questo progetto e ad avermi dedicato il suo prezioso tempo.

Inoltre, voglio ringraziare i miei amici e colleghi universitari, in particolare modo Riccardo, Marco e Filippo, per aver condiviso con me questi bellissimi tre anni passati insieme all'insegna di burle e di serenità.

Un grazie speciale va anche ai miei amici da una vita, con i quali sono cresciuto ed ho trascorso tantissimi bei momenti sin dall'infanzia, i primi ad esserci nei momenti difficili, sempre i primi ad esserci in quelli di festa.

Un ringraziamento sentitissimo ai miei genitori Mauro e Daniela che mi hanno sempre assecondato nelle scelte ed alle volte indicato la strada che ritenevano più opportuna. È in gran parte grazie a loro se oggi ho raggiunto questo importante traguardo.

Un ultimo, ma non meno importante, grazie ad Elisa, la mia cara Elisa, che mi ha, come pochi altri, supportato, che mi ha insegnato tanto, tantissimo anche se, forse, lei non lo sa.

Per concludere, grazie a me, alla mia pazienza ed alla mia tenacia.

Bibliografia

[1] Frank Vahid, Tony Vargis, Bailey Miller (2012), *Programming Embedded Systems: An introduction to Time-Oriented Programming.*

[2] James F. Kurose, Keith W. Ross, *Reti di calcolatori e internet: un approcio top-down*, Pearson.

[3] Nicholas R. Jennings (2001), *An agent-based approach for building complex software systems.*

[4] Paul Viola, Michael Jones (2001), *Rapid Object Detection using a Boosted Cascade of Simple Features.*

Sitografia

[5] Documentazione Libreria OpenCV,
https://docs.opencv.org/

[6] Flir Lepton 3.5 Datasheet,
https://www.flir.com/globalassets/
imported-assets/document/lepton-3-3.5-datasheet.pdf

[7] Pylepton,
https://github.com/groupgets/pylepton