UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

I QR Code e la loro evoluzione

Relatore:

Prof. Stefano Tomasin

Laureando:

Giacomo Avanzi

MATRICOLA N. 1223262

ANNO ACCADEMICO 2021/2022

Data di laurea 20/09/2022

Abstract

I QR Code sono codici a barre bidimensionali capaci di immagazzinare una grande quantità di dati, caratterizzati da un'elevata robustezza agli errori e da una lettura veloce, possibile da qualsiasi angolazione e senza allineamento. In questo lavoro di tesi si analizzano i processi di codifica e decodifica dei QR Code, interpretati come parte di un sistema di telecomunicazione. In questa analisi dei QR Code si fa riferimento alle specifiche regolate dallo Standard ISO/IEC 18004:2015. In seguito, si indaga lo stato dell'arte dei QR Code, focalizzandosi sulle nuove tecniche per aumentare la capacità dei codici, riguardo l'uso di diversi spazi colore nella rappresentazione dei moduli (modulazione), la fusione (*multiplexing*) di più codici attraverso simboli o colori e i metodi di compressione dei dati, discutendo vantaggi e svantaggi di ciascuna proposta. Nella parte sperimentale si considerano tre differenti dataset di QR Code standard, contenenti immagini in cui i QR Code sono interessati da diverse tipologie di distorsione (rumore) introdotte dal canale di trasmissione, quali illuminazione irregolare, rotazione, deformazione prospettica e sfocatura. Utilizzando il decodificatore *pyzbar*, si effettua un'analisi delle prestazioni del processo di decodifica.

Ai miei genitori e a mio fratello.

Indice

Introduzione

L'obiettivo di questo lavoro di tesi è l'analisi dei *Quick Response (QR) Code*, codici a barre bidimensionali composti da una matrice quadrata contenente moduli chiari e scuri, in cui può essere codificata una vasta quantità di dati, fino a 7089 caratteri numerici. I QR Code sono largamente diffusi oggigiorno negli ambiti più disparati: a partire dai pagamenti elettronici fino agli annunci pubblicitari, toccando anche l'ambito della salute. Alla base di questa grande diffusione ci sono anche motivi tecnici, tra cui spiccano l'elevata capacità di cui i codici sono dotati, la possibilità di lettura da qualsiasi angolazione e senza alcun particolare allineamento, e la correzione degli eventuali errori introdotti da distorsioni che avvengono nei casi reali di utilizzo, quali ad esempio danneggiamento del simbolo oppure condizioni luminose anomale o distorsioni prospettiche durante la fase di acquisizione dell'immagine.

La prima parte dell'elaborato consiste, innanzitutto, nel Capitolo 1 in cui viene introdotto il contesto che ha dato origine a questi codici, passando poi in rassegna i motivi alla base del grande successo e della diffusione su larga scala, discutendo vantaggi e svantaggi, anche legati ai tradizionali codici a barre monodimensionali. Il Capitolo 2 presenta una spiegazione dettagliata dei processi chiave di codifica e decodifica: la fase di codifica, a partire dai dati in ingresso, porta alla generazione del QR Code, adottando un'opportuna codifica di sorgente (4 modalità diverse), codifica di canale (codici di Reed-Solomon) e modulazione (quadrati bianchi e neri, detti moduli), dall'altro lato la fase di decodifica, a seguito della fase di thresholding, ha l'obiettivo di riottenere i dati codificati partendo dal QR Code. Nella conclusione del capitolo si conduce un parallelismo tra le operazioni di codifica e decodifica dei QR Code e le operazioni compiute dai sistemi di telecomunicazione.

In seguito, si procede trattando l'evoluzione dei codici nel tempo: in particolare, nel Capitolo 3 si esplora lo Standard ISO/IEC 18004:2015 che regola le specifiche dei QR Code, tra cui i parametri dei codici Reed-Solomon, poi nel Capitolo 4 si presenta lo stato dell'arte delle tecniche per estendere la capacità di immagazzinamento dei dati. Nello specifico vengono trattati i seguenti metodi: l'uso di diversi spazi colore nella rappresentazione dei moduli (modulazione),

la fusione (*multiplexing*) di più codici attraverso simboli o colori e i metodi di compressione dei dati, di ciascuno si analizzano i relativi vantaggi e svantaggi.

Nella seconda parte dell'elaborato, nel Capitolo 5, si propone un'analisi sperimentale di 3 diversi dataset di QR Code, per determinare l'effetto sulla fase di decodifica delle distorsioni (rumore) introdotte dal canale di trasmissione, considerando quelle più verosimili nei casi d'uso dei codici. Le immagini ritraggono singoli (dataset 1, dataset 2 e prima porzione del dataset 3) o multipli (seconda porzione del dataset 3) QR Code, inseriti in ambienti reali eterogenei, che codificano dati differenti e appartengono a versioni diverse. Le distorsioni presenti nei codici consistono in illuminazione irregolare, rotazione, deformazione prospettica e sfocatura: esse presentano diversi livelli di intensità e possono essere applicate singolarmente o in contemporanea ad altre. Dai risultati ottenuti in ciascun dataset, nei casi in cui non è avvenuta la decodifica si discute riguardo alle distorsioni e agli aspetti che esse hanno compromesso maggiormente, in merito soprattutto alla fase di demodulazione e alla robustezza della struttura dei QR Code.

Capitolo 1

Storia e successo dei QR Code

1.1 Origini

La nascita dei QR Code risale all'ultimo decennio del 1900, in Giappone, in risposta alle esigenze dell'epoca, come descritto in [7].

Durante gli anni '80 c'è un grande utilizzo dei codici a barre in molti ambiti, dalla manifattura fino alla logistica e alla vendita al dettaglio. Tuttavia, all'inizio degli anni '90, nonostante la grande diffusione, emergono i limiti di questo strumento: in particolare il vincolo più grande è il numero massimo di caratteri alfanumerici che i codici a barre potevano conservare, fissato a 20. Infatti le industrie manifatturiere iniziano a richiedere codici con capacità maggiori, a causa di un'evoluzione del modello produttivo verso una produzione più flessibile e con un controllo e tracciamento più accurato.

La Denso Wave Incorporated (poi una compagnia di Denso Corporation), che al tempo sviluppava i lettori per i codici a barre, accoglie la richiesta degli utenti. Così, incoraggiata da questo nuovo progetto, l'azienda affida a un team guidato da Masahiro Hara, composto da soli due membri, lo sviluppo di un nuovo codice bidimensionale con prestazioni maggiori.

A differenza dei tradizionali codici a barre, dove le informazioni sono codificate e disposte lungo una sola direzione, i codici bidimensionali offrono una disposizione più ampia, sia lungo la direzione orizzontale che verticale: in altri termini si può pensare a dati posti nelle colonne e nelle righe di una matrice. Un altro obiettivo desiderato, oltre alla maggiore capacità, è quello di rendere la lettura di questi nuovi codici quanto più veloce possibile: infatti la localizzazione del codice bidimensionale da parte dello scanner è più difficoltosa rispetto a quella del codice a barre.

Dopo un anno e mezzo di sviluppo, il team produce il primo QR Code capace di contenere

fino a 7000 cifre e supportare anche i caratteri Kanji. Questo nuovo codice, grazie alla struttura adottata in fase di sviluppo, può conservare una grande quantità di informazioni e, allo stesso tempo, può essere letto in una velocità dieci volte maggiore rispetto agli altri codici.

Nel 1994, Denso Wave annuncia la diffusione del proprio primo QR Code, per esteso, *Quick Response Code*. Il nome viene scelto proprio per mettere in luce l'elevata velocità nella decodifica del contenuto del nuovo codice.

Nippondenso Co. Ltd., nome originale con cui viene fondata la Denso Wave, registra e detiene il brevetto giapponese *JPH07254037A* dal 03/10/1995 e l'analogo europeo *EP0672994A1* dal 20/09/1995, in cui il primo inventore risulta Masahiro Hara.

1.2 Utilizzo e campi di applicazione

Dato che Denso Wave è una sussidiaria del gruppo Toyota,[1] all'inizio i QR Code sono impiegati per tracciare i componenti di automobili nelle fabbriche. Questo ha contribuito a rendere più efficiente la gestione di diversi compiti durante la filiera, a partire dalla produzione fino alle spedizioni.

La Denso Wave, vedendo il rapido diffondersi dell'utilizzo della propria invenzione, decide di renderne pubbliche le specifiche [8] in modo tale da permettere a tutti di poter creare e utilizzare liberamente i QR Code. Tuttavia ne mantengono la proprietà intellettuale, senza però esercitarla.

In seguito, vista la richiesta da parte della società di rendere trasparenti i processi di produzione industriale relativamente alla tracciabilità dei prodotti, l'uso di questo strumento si estende anche al dì fuori del contesto che l'ha partorito. Le compagnie farmaceutiche, alimentari e di produzione di lenti a contatto adottano i Qr Code per controllare la gestione della loro merce. In particolare le industrie alimentari, sospinte dalla domanda dei consumatori di trasparenza riguardo la filiera degli alimenti, hanno implementato questi codici nell'intero processo produttivo e logistico. In questo modo i QR Code diventano uno strumento fondamentale per contenere i dati relativi ai processi di produzione delle aziende, appartenenti a molti settori diversi.

Come sostenuto in [12], attualmente i QR Code sono diffusi in tutti gli ambiti in cui erano usati i tradizionali codici a barre, ad esempio la manifattura, la vendita al dettaglio, il settore medico e la logistica. Ma non solo, questi codici trovano spazio anche in nuovi campi di

[1] Toyota Motor Corporation è una multinazionale giapponese che produce autoveicoli, fondata nel 1933 a Toyota da Kiichirō Toyoda

Nazione	Utenti (mln)	Variazione % 2012-2011	% sul totale utenti smartphone
EU5	12.390	96%	14.1%
Francia	2.843	71%	12.5%
Germania	5.084	128%	18.6%
Italia	2.765	75%	11.9%
Spagna	3.381	218%	16.0%
Regno Unito	3.316	43%	11.4%

Tabella 1.1: Utenti smartphone che hanno scansionato dei QR Code, media mensile nel trimestre culminante in Luglio 2012 vs Luglio 2011 (fonte: ComScore)

	EU5	Francia	Germania	Italia	Spagna	Regno Unito
Informazioni su prodotti	71.7%	65.4%	77.9%	69.2%	71.1%	70.1%
Informazioni su eventi	31.8%	32.7%	28.9%	36.4%	36.5%	27.0%
Dettagli su progetti sociali	12.1%	9.5%	10.0%	18.6%	13.4%	10.8%
Buoni/Offerte	19.4%	20.3%	19.6%	17.6%	22.2%	17.0%
Download di app	13.4%	17.5%	11.4%	14.8%	13.7%	11.2%

Tabella 1.2: Risultati delle scansioni dei QR Code da smartphone, media mensile nel trimestre culminante in Luglio 2012 (fonte: ComScore)

applicazione, ovvero il commercio sui dispositivi mobili, pubblicità online, ticket elettronici, pagamenti telematici, identificazione, insegnamento accademico, etc.

1.3 Diffusione

In pochissimo tempo i QR Code entrano nello stile di vita della società moderna, diventando uno strumento utilizzato ogni giorno da una grande quantità di persone. Un contributo importante a questa diffusione proviene dal grande sviluppo del mercato degli smartphone e della connessione internet mobile.

ComScore, una compagnia che gestisce campagne mediatiche riguardo diversi settori commerciali su numerose piattaforme, ha stilato un report [4] nel Settembre del 2012, in cui ha analizzato l'uso degli smartphone in correlazione alla scansione dei QR Code nei mercati dei primi cinque paesi europei, ovvero Francia, Germania, Italia, Spagna e Regno Unito. Come si può vedere in Tabella 1.1, dallo studio è emerso che, in media, nel trimestre culminante in Luglio 2012, 17.4 milioni di utenti (cioè il 14.1% degli utenti smartphone) con il proprio smartphone hanno scansionato dei QR Code, con un incremento del 96% rispetto allo stesso periodo dell'anno precedente. Di conseguenza, nonostante la diffusione degli smartphone e le connessioni internet mobili fossero ancora limitate, la presenza dei QR Code sul mercato iniziava ad affermarsi. Inoltre, ComScore mette in evidenza che circa 3 scansioni su 4 hanno lo scopo di reperire informazioni riguardanti prodotti (Tabella 1.2), quindi questo rappresenta l'uso prin-

cipale in Europa verso Luglio 2012. Altri utilizzi diffusi servono per ottenere informazioni su eventi o scaricare coupon e offerte.

1.4 Vantaggi e Svantaggi dei QR Code

1.4.1 Vantaggi

I QR Code hanno avuto fin da subito un grande successo grazie principalmente a due aspetti, cardine nel relativo processo di sviluppo: l'elevata capacità di immagazzinamento dei dati e la rapidità di lettura. Infatti un singolo QR Code può contenere fino a 7089 caratteri numerici, in uno spazio veramente ridotto. Inoltre, i caratteri codificabili non si limitano solo a numeri, ma ci sono anche i caratteri alfanumerici, di un byte e Kanji. [12]

La lettura di questi simboli è altrettanto significativa, dato che possono essere scansionati e decodificati velocemente da qualsiasi angolazione, senza richiedere alcun tipo di allineamento. Per fare ciò si sfrutta la sofisticata struttura con cui sono stati progettati: più precisamente i finder pattern aiutano nell'identificazione del simbolo, mentre gli alignement pattern (se presenti) correggono la distorsione (cfr. Sezione 3.2.1). [13]

Anche le tecniche per la correzione degli errori usate nei QR Code assumono importanza, visto che permettono il recupero di una percentuale dei dati conservati, in base a quattro livelli di correzione, fino al 30%. Quindi, se il simbolo presenta distorsioni perché è danneggiato o sporco, oppure risulta incompleto, o ancora ci sono condizioni luminose inadeguate, può essere ugualmente decodificato, nei limiti della capacità correttiva del codice usato. [12]

Infine, il rilascio al pubblico dominio delle specifiche del QR Code da parte della Denso Wave è stato fondamentale nel successo di questi simboli: essi sono disponibili liberamente per chiunque. Ciascuno, comprese le aziende, può creare e utilizzare un QR Code per gli utilizzi più diversi, in qualsiasi ambiente, come visto nella Sezione 1.2.

1.4.2 Svantaggi

Nonostante i QR Code abbiano molti vantaggi, ci sono alcuni aspetti negativi che vanno presi in considerazione.

Il primo principale svantaggio è l'assenza di anteprima del contenuto nel QR Code [13]: solo dopo aver scansionato il simbolo e decodificato i dati presenti si risale alla risorsa puntata. Se la risorsa è una pagina web malevola, viene compromessa la sicurezza dell'utente. Ad esempio può essere avviato lo scaricamento di un file dannoso nel dispositivo, oppure si può essere

ridirezionati su una pagina contenente campi da completare con informazioni sensibili. Esistono delle applicazioni per dispositivi mobili, ovvero appositi lettori, che risolvono questo problema, permettendo di avere un'anteprima del QR Code prima di aprire la pagina della risorsa sul browser.

Un altro svantaggio riguarda la necessità di possedere obbligatoriamente un lettore, installato nel dispositivo dell'utente, per decodificare il QR Code. Questo limita la platea di utilizzatori, soprattutto coloro che non sono abili nell'utilizzo dei nuovi mezzi tecnologici. [12]

1.4.3 Confronto QR Code - Codici a barre

I codici a barre sono codici monodimensionali, in cui le informazioni sono disposte lungo una sola direzione. Sono composti da elementi verticali scuri, le barre, e da elementi chiari che separano le barre, gli spazi. L'informazione può essere contenuta solo nelle barre (codici a barre discreti) oppure sia nelle barre che negli spazi (codici a barre continui). Invece, come già detto, i QR Code sono codici bidimensionali, composti da una matrice, nella quale le informazioni sono riportati in direzione verticale e orizzontale nei moduli chiari e scuri.

In [10] si propone un confronto tra codici a barre e QR Code, basato sulle seguenti caratteristiche:

1. Capacità massima
    - Codice a barre: 10-20 caratteri numerici
    - QR Code: 7089 caratteri numerici (Versione 40-L)
2. Tipi di dati codificabili
    - Codice a barre: caratteri numerici e alfanumerici
    - QR Code: caratteri numerici, alfanumerici, di un byte e Kanji
3. Modalità di lettura
    - Codice a barre: lettura allineata orizzontalmente
    - QR Code: lettura più veloce, senza allineamento e da qualsiasi angolazione, grazie ai pattern funzionali
4. Rilevazione e correzione degli errori
    - Codice a barre: sono presenti dei simboli di controllo, ma non è permessa la lettura in caso di danneggiamento del codice

- QR Code: è permessa la lettura anche in caso di danneggiamento, correggendo gli errori e recuperando fino al 30% del contenuto del codice

5. Spazio occupato

   - Codice a barre: sono rappresentati 10 caratteri numerici in uno spazio di circa 50 mm x 20 mm
   - QR Code: sono rappresentati 40 caratteri numerici in uno spazio di circa 5 mm x 5 mm

Si possono notare i grandi cambiamenti introdotti dai QR Code, principalmente nella capacità di immagazzinamento, nel processo di lettura e nella correzione degli errori. Questi elementi, di conseguenza, hanno portato a una struttura e a un processo di codifica e decodifica più complesso rispetto ai codici a barre.

Capitolo 2

Codifica e decodifica dei QR Code

Il Quick Response (QR) Code è un codice a barre bidimensionale, cioè a matrice, composto da moduli bianchi e neri disposti su una matrice quadrata. Il sistema di elaborazione dei QR Code si basa su due processi fondamentali: la fase di codifica e la fase di decodifica.

La codifica riceve i dati, quali, ad esempio, testo o indirizzi web (URL), e porta alla generazione del codice a barre bidimensionale. Il codificatore ha il compito di codificare i dati e produrre il QR Code.

La decodifica è il processo inverso, ovvero a partire dal QR Code recupera i dati immagazzinati. Il decodificatore (lo scanner) ha il compito di decodificare il simbolo e recuperarne i dati.

2.1 Codifica

Il processo di codifica converte i dati di input in un QR Code, procedendo in ordine attraverso i seguenti passi [12]. La rappresentazione tramite un diagramma di flusso viene mostrata in Fig. 2.2.

1. Analisi dei dati

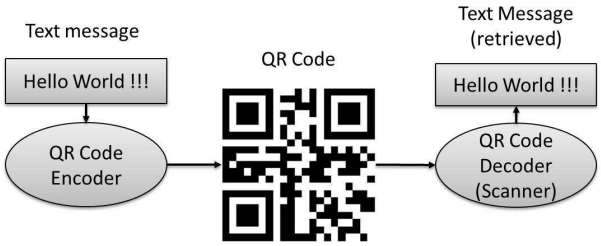


Figura 2.1: Panoramica della codifica e della decodifica

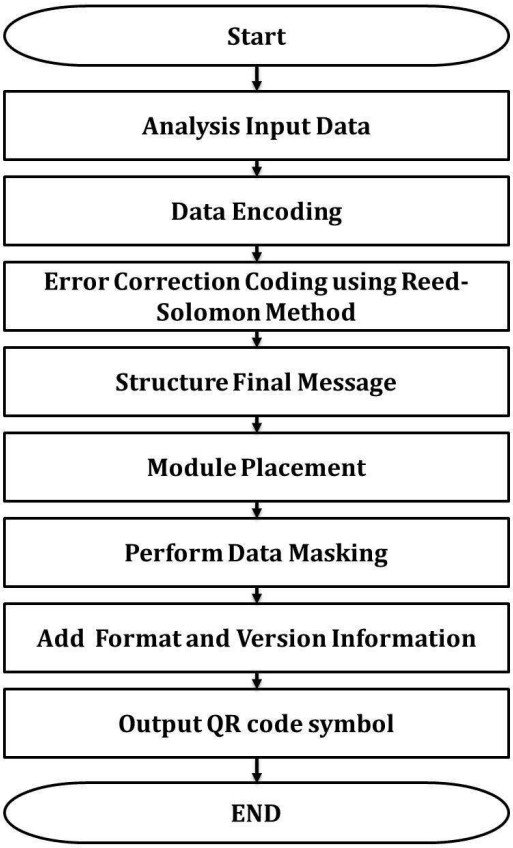


Figura 2.2: Processo di codifica [12]

La stringa dei dati da codificare contiene diversi tipi di caratteri, è quindi necessario analizzare i dati per identificare il tipo corretto e scegliere il metodo adatto con cui convertire le informazioni in stringa di bit. Il QR Code supporta quattro modalità differenti di codifica (cfr. Sezione 3.3.1): numerica, alfanumerica, byte e Kanji. Ciascuna di esse utilizza una procedura diversa per convertire un determinato tipo di caratteri in bit, ed è ottimizzata per generare la stringa binaria più corta possibile per lo specifico tipo di dato.

2. Codifica dei dati

Prima di procedere con la fase di codifica, si deve scegliere il livello di correzione degli errori e la più piccola dimensione del QR Code adatta per i dati di input. I QR Code usano i codici Reed-Solomon per la correzione degli errori, ci sono quattro livelli di correzione possibili (cfr. Sezione 3.1), dal più basso al più elevato: L, M, Q e H. I livelli maggiormente correttivi richiedono più bit, di conseguenza aumenta la dimensione del QR Code. Le dimensioni diverse dei QR Code sono dette versioni (cfr. Sezione 3.1), la più piccola è la versione 1, la più grande è la versione 40. Ogni versione ha una capacità massima, che dipende dalla modalità di codifica e dal livello di correzione degli errori adottati.

Successivamente, la stringa binaria che conterrà i dati da codificare deve iniziare con l'indicatore (4 bit) della modalità di codifica scelta al punto precedente, e subito dopo l'indicatore (lunghezza in bit variabile in base alla versione e alla modalità di codifica) del contatore di caratteri da codificare. A questo punto possono essere concatenati alla stringa i bit ottenuti dalla specifica modalità di codifica applicata ai dati di input. Come risultato si ha un nuovo segmento, che inizia con il primo bit (più significativo) dell'indicatore della modalità di codifica e termina con l'ultimo bit (il meno significativo) della stringa binaria dei dati codificati. Possono esserci più segmenti, a seconda del numero di sequenze di dati da codificare: in questo caso le stringhe binarie ottenute da ogni segmento sono concatenate insieme nell'ordine. Alla fine del segmento, o della sequenza di segmenti, viene posto un terminatore di quattro zeri, che può risultare più corto o mancante in relazione alla capacità rimanente del QR Code.

Infine, la stringa dei dati contenente tutti i segmenti codificati viene suddivisa in simboli, ciascuno di 8 bit, chiamati nel seguito *data codeword* (in accordo con lo Standard ISO 18004:2015). Può essere necessario aggiungere alcuni bit, o intere data codeword, di padding (cfr Sezione 3.3.3).

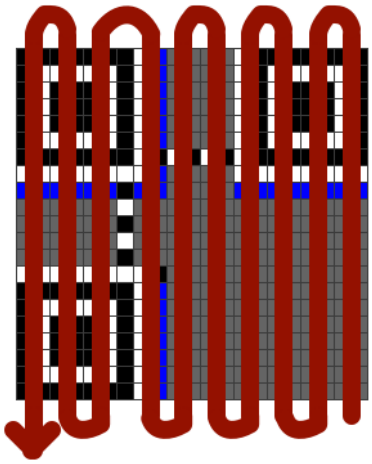


Figura 2.3: Progressione nella disposizione dei bit sulla matrice (Thonky. URL: https://www.thonky.com/qr-code-tutorial/data-bit-progression.png (visitato il 08/07/2022).)

3. Codifica dei simboli per la correzione degli errori

   I QR Code adottano i codici Reed-Solomon per la correzione degli errori (cfr. Sezione 3.4.1). La sequenza di data codeword, suddivisa in blocchi più piccoli se il QR Code è di versione superiore alla 2, viene processata dagli algoritmi dei codici di correzione degli errori. In questo modo, a partire da ciascun blocco di data codeword, applicando i codici Reed-Solomon, si ottengono i blocchi contenti i simboli di parità[1] per la correzione degli errori (cfr. sezione 3.4.2). Il numero dei blocchi di data codeword coincide con il numero di blocchi dei simboli per la correzione degli errori.

4. Costruzione della sequenza finale con le data codeword e i simboli di parità

   Una volta ottenuti, per ciascun blocco, i simboli di parità, essi devono essere arrangiati in modo opportuno [1] con le data codeword. Prima vengono disposte le data codeword, poi i simboli per la correzione degli errori.

   In particolare la disposizione delle data codeword nella sequenza di bit finale deve essere la seguente:

   (a) la prima data codeword dal primo blocco;

   (b) la prima data codeword dal secondo blocco;

   (c) la prima data codeword dal terzo blocco;

   (d) nell'ordine le prime data codeword da tutti i blocchi rimanenti;

   (e) la seconda data codeword dal primo blocco;

[1] Nello Standard ISO/IEC 18004:2015 si fa riferimento a parole di codice per la correzione degli errori (*error correction codeword*)

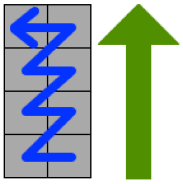


Figura 2.4: Disposizione verso l'alto dei bit nella colonna (Thonky. URL: https://www.thonky.com/qr-code-tutorial/upward.png (visitato il 08/07/2022).)

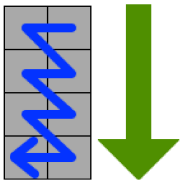


Figura 2.5: Disposizione verso il basso dei bit nella colonna (Thonky. URL: https://www.thonky.com/qr-code-tutorial/downward.png (visitato il 08/07/2022).)

(f) nell'ordine le seconde data codeword da tutti i blocchi rimanenti;

(g) lo stesso pattern per tutte le data codeword da tutti i blocchi rimanenti.

Dopo aver fatto ciò, si procede seguendo lo stesso pattern per i simboli di parità, finché non vengono esauriti. Il motivo alla base di questa procedura, detta *interleaving* (lett. interfogliamento), sta nel ridurre la probabilità che errori consecutivi al QR Code superino la capacità correttiva del singolo blocco.

Per alcune versioni di QR Code, nel caso in cui la stringa finale non abbia una lunghezza pari a quella richiesta dalla specifica, è necessario aggiungere un certo numero di bit finali a 0, detti remainder bit.

5. Disposizione dei moduli sulla matrice

La stringa di bit ottenuta al punto precedente va inserita nella matrice quadrata del QR Code, composta da numerosi quadrati, detti moduli. Si preferisce il termine modulo a pixel per non creare confusione con i pixel visualizzati a schermo: ad esempio la versione 1 dei QR Code è composta sempre da 21 x 21 moduli, indipendentemente che sullo schermo occupi 42 x 42 pixel, o 84 x 84 pixel. Tuttavia, sulla matrice non sono presenti solo i dati codificati, ci sono anche alcuni elementi strutturali che non corrispondono a dati, detti pattern funzionali. Sono necessari per la corretta localizzazione del QR Code e del relativo orientamento, saranno trattati in modo più approfondito nel capitolo successivo.

I bit dei dati vanno disposti sulla matrice partendo dall'angolo in basso a destra e procedendo verso l'alto in una colonna ampia 2 moduli. I bit a 0 corrispondono a moduli

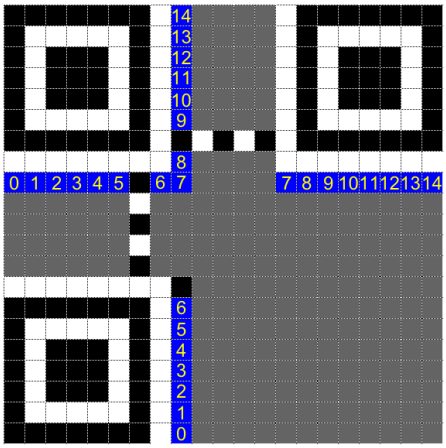


Figura 2.6: Posizioni dell'informazione sul formato (Thonky. URL: https://www.thonky.com/qr-code-tutorial/format-layout.png (visitato il 08/07/2022).)

di colore bianco, i bit a 1 corrispondono a moduli di colore nero. Quando si raggiunge il bordo superiore della matrice, si occupa la colonna successiva a sinistra e si continua disponendo i bit verso il basso. Ogni volta che si raggiunge il bordo della matrice, si passa alla colonna successiva, di ampiezza pari a 2 moduli, e si cambia alternativamente la direzione seguita, come mostrato in Fig.2.3.

La disposizione dei bit avviene solitamente nella direzione verticale, in blocchi di moduli 2 x 4. All'interno di una colonna i bit sono arrangiati seguendo un andamento da destra a sinistra, verso l'alto (Fig 2.4) o verso il basso (Fig. 2.5) in base alla direzione seguita all'interno della colonna. Il bit più significativo di ogni data codeword o simbolo di parità deve essere posizionato nel primo modulo disponibile.

Nel riempimento della matrice, quando si incontrano dei pattern funzionali, si devono saltare i moduli occupati, scegliendo i moduli inutilizzati più vicini nella direzione in cui si sta procedendo. Un'unica eccezione a questa regola va considerata quando si raggiunge un particolare pattern funzionale, il timing pattern verticale: si deve sempre riempire la colonna successiva a sinistra, in modo che nessuna colonna vada a sovrapporsi con il timing pattern.

6. Applicazione di un pattern di mascheramento

   Una volta che i moduli sono stati arrangiati, alla matrice viene applicato un mascheramento, con lo scopo di modificare il QR Code per renderlo più facilmente scansionabile

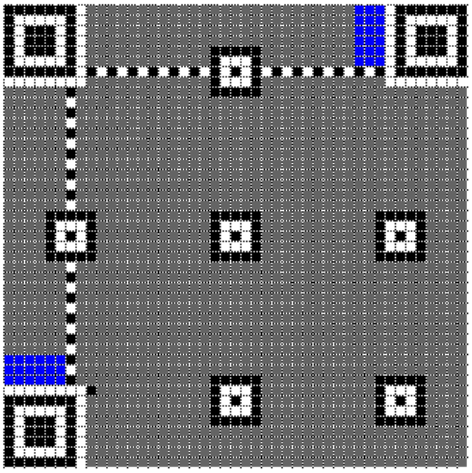


Figura 2.7: Posizioni dell'informazione sulla versione (Thonky. URL: https://www.thonky.com/qr-code-tutorial/version-location-1.png (visitato il 08/07/2022).)

da un lettore. In questo modo vengono rimosse grandi zone di soli moduli chiari o scuri, oppure disposizioni di moduli tali da confondersi con pattern funzionali. L'operazione di mascheramento consiste in uno XOR tra i bit nella matrice (modulo chiaro = bit a 0, modulo scuro = bit a 1) e i bit di una matrice con le stesse dimensioni (maschera), con tutti i bit nulli, tranne alcuni in posizioni che seguono un preciso pattern, detto pattern di mascheramento, tale da soddisfare una determinata formula. In altre parole vengono invertiti determinati moduli originali, secondo un preciso pattern.

Esistono otto varianti di maschere, ciascuna con la relativa formula per identificare quali bit devono assumere valore 1. Tali maschere vanno applicate solamente ai moduli per i dati e per la correzione degli errori, vanno esclusi i pattern funzionali e le aree riservate al formato e alla versione sulla matrice. Per scegliere quale pattern di mascheramento applicare è necessario provarli applicandoli tutti, valutare ciascun caso singolarmente e attribuire un punteggio di penalità in base ad alcune regole. Il pattern che riceverà la penalità totale più bassa sarà quello effettivamente applicato. Per ulteriori dettagli sui pattern di mascheramento si rimanda alla Sezione 3.3.4.

7. Aggiunta delle informazioni sul formato e sulla versione

   Lo step finale richiede di aggiungere nel QR Code le informazioni sul formato e, se necessario, sulla versione.

   - L'informazione sul formato è composta da 15 bit, in cui i 5 bit più significativi specificano rispettivamente dal bit più significativo il livello di correzione degli errori

e il pattern di mascheramento adottati, gli altri 10 bit servono per correggere gli errori (Codice BCH (15, 5)) sui bit precedenti. Le specifiche richiedono che i 15 bit risultanti siano sottoposti a un'operazione di XOR con una particolare stringa di mascheramento (cfr. Sezione 3.2.2). I bit relativi al formato sono riportati sul QR Code nelle zone mostrate in Fig. 2.6, la disposizione dei bit richiede nella posizione 0 il bit meno significativo. Appaiono ripetuti due volte in modo tale da fornire ridondanza, poiché tali dati sono necessari per la decodifica completa del simbolo.

- L'informazione sulla versione è necessaria nei QR Code di versione 7 o superiore. Essa è composta da 18 bit, in cui i 6 bit più significativi contengono la versione del simbolo, gli altri 12 servono per correggere gli errori (Codice di Golay (18, 6)) sui bit precedenti. I bit in questione vengono riportati sul QR Code nelle zone mostrate in Fig. 2.7, la disposizione dei bit richiede nell'angolo in alto a sinistra il bit meno significativo, procedendo poi da sinistra a destra e sulle righe inferiori. Anche questi dati sono ripetuti due volte per fornire ridondanza, poiché tali dati sono necessari per la decodifica completa del simbolo.

2.2 Decodifica

La decodifica è il processo inverso della codifica, dalla lettura del QR Code ha il compito di restituire i dati. Essa avviene attraverso dei passi speculari a quelli esposti in precedenza. Nella Fig. 2.8 viene riportato il diagramma di flusso corrispondente.

1. Riconoscimento dei moduli

   La localizzazione del QR Code avviene attraverso l'identificazione dei tre finder pattern e degli eventuali alignment pattern, poi si stabilisce la dimensione nominale del codice e, provvisoriamente, la versione. Ma prima di procedere con questi passi, si deve determinare quando un modulo è scuro o chiaro basandosi su un'opportuna soglia ricavata dall'immagine. Quindi viene compiuta un'operazione di *thresholding* (lett. sogliatura) a partire dall'immagine del codice in scala di grigi, per segmentarla in base all'intensità dei pixel, ovvero ogni pixel viene confrontato con il valore soglia e si decide se sostituirlo con un pixel bianco o nero, ottenendo così un'immagine binaria. Si possono applicare diversi algoritmi differenti in base all'adozione di una soglia fissa, adattiva o locale, oppure multipla: il più semplice a soglia fissa, ad esempio, adotta la media tra la minima e la massima intensità dei pixel nell'immagine in scala di grigi.

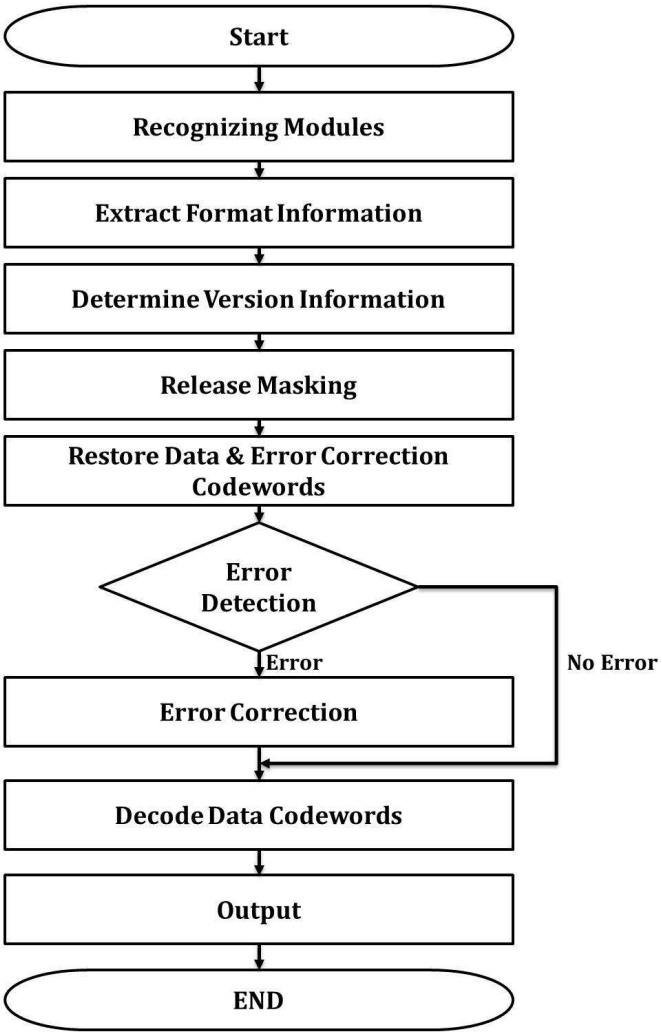


Figura 2.8: Processo di decodifica [12]

Infine, dall'immagini binaria si ricava la matrice di bit in cui i moduli scuri sono mappati con un bit a 1 e i moduli chiari con un bit a 0.

2. Estrazione dell'informazione sul formato

   Si decodifica l'informazione sul formato dalla zona adiacente al separatore del finder pattern nell'angolo superiore sinistro, ottenendo così il livello di correzione degli errori e il pattern di mascheramento applicato al simbolo. Se gli errori sull'informazione sul formato superano la capacità correttiva del codice BCH, si applica la stessa procedura alla zona di ridondanza disposta in parte sotto il separatore del finder pattern in alto a destra, e in parte a destra del separatore del finder pattern in basso a sinistra. Se ancora non è possibile ottenere una stringa valida per il formato, si prova una lettura nella direzione opposta e una decodifica sull'immagine speculare, con le righe e le colonne trasposte della matrice.

3. Estrazione dell'informazione sulla versione

   Se il QR Code è di versione 7 o superiore, deve essere determinata l'informazione sulla versione analizzando la zona a sinistra del separatore del finder pattern superiore destro, oppure la zona di ridondanza sopra il separatore del finder pattern inferiore sinistro. Se gli errori eccedono la capacità correttiva del codice di Golay, si considera la versione calcolata provvisoriamente nella fase iniziale, ovvero allo stesso modo di come avviene per i QR Code di versione 6 o inferiore, che non possiedono tale informazione.

4. Rimozione del pattern di mascheramento

   Viene effettuata l'operazione di XOR tra i bit della regione di codifica, cioè esclusi gli elementi che non contengono dati (es. pattern funzionali), e i bit del pattern di mascheramento, identificato in precedenza dall'informazione sul formato. Di conseguenza viene tolto l'effetto del processo di mascheramento dei dati applicato in fase di codifica.

5. Recupero delle data codeword e dei simboli di parità

   A partire dalla matrice, viene ricostruita la sequenza di bit originale seguendo l'ordine inverso rispetto a quello con cui i bit sono stati disposti in fase di codifica (cfr. passo 5 nella Sezione 3.1). Inoltre, dalla stringa ottenuta si deve rimuovere l'interleaving tra le data codeword e i simboli di parità, seguendo al contrario la procedura riportata nel passo 4 nella sezione Codifica. In questo modo sono recuperate tutte le data codeword e tutti i simboli per la correzione degli errori.

6. Rilevamento e correzione degli errori

   Si utilizzano i simboli di parità per rilevare e correggere eventuali errori sui dati, compatibilmente con la massima capacità correttiva dei codici Reed-Solomon, data dal livello di correzione degli errori specificato nell'informazione sul formato.

7. Decodifica delle data codeword

   Le data codeword, eventualmente suddivise in blocchi, vengono riassemblate in un'unica sequenza binaria. Questa stringa di bit viene ridotta in segmenti: ciascuno dei quali inizia con l'indicatore della modalità di codifica usata (4 bit) e successivamente l'indicatore del contatore di caratteri codificati (lunghezza in bit variabile in base alla versione), seguiti poi dai bit dei dati codificati. Infine, ogni segmento viene decodificato secondo la modalità usata in fase di codifica, esplicitata dall'indicatore, riottenendo così i dati originali.

2.3 Parallelismo con i sistemi di telecomunicazione

I sistemi di trasmissione sono composti da tre entità:

- sorgente
- canale
- destinazione

La sorgente ha il compito di trasmettere informazioni verso la destinazione, attraverso un canale fisico rumoroso. Nelle trasmissioni digitali la sorgente emette un segnale, analogico (audio, video, immagini) o discreto (testo, bit), che viene prima, se necessario, convertito in formato digitale a tempo discreto e valori discreti (Analog-to-Digital Conversion ADC) e poi mappato in una rappresentazione binaria (codifica). Questo intero processo prende il nome di digitalizzazione. Per la successiva trasmissione sul canale fisico, viene effettuata un'operazione di modulazione in cui si converte l'informazione binaria in forma analogica, ovvero si associano ai gruppi di bit segnali analogici (elettrici, elettromagnetici, ottici a secondo del mezzo) di opportune forme d'onda, inviati poi nel canale fisico di trasmissione. Al ricevitore si eseguono le stesse operazioni in modo complementare: demodulazione, decodifica di canale e decodifica di sorgente. Se necessaria, infine, avviene la conversione del segnale digitale in analogico (Digital-to-Analog Conversion DAC).

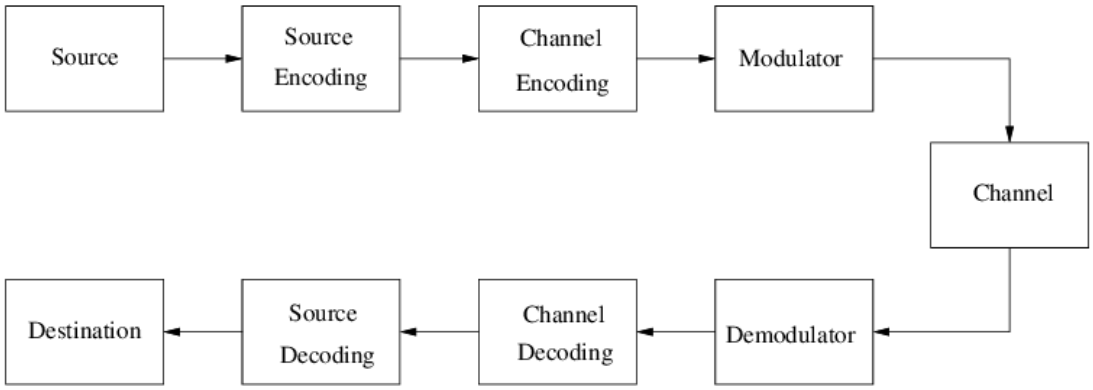


Figura 2.9: Architettura di un sistema di telecomunicazione (W. Zhang *et al.* "Learning-Aided Unary Error Correction Codes for Non-Stationary and Unknown Sources". In *IEEE Access* 4 (feb. 2016). DOI: 10.1109/ACCESS.2016.2544060)

In particolare, prima che il segnale digitale giunga al modulatore, subisce due elaborazioni fondamentali: la codifica di sorgente e la codifica di canale.

La codifica di sorgente è un'operazione effettuata per eliminare la ridondanza dei dati: viene eseguita una compressione, con perdite (lossy) o senza perdite (lossless), con lo scopo di ridurre la dimensione dei dati da inviare e facilitare quindi la trasmissione. La codifica di canale aggiunge informazioni di controllo in base allo specifico canale di comunicazione rumoroso per fornire robustezza e garantire l'integrità dei dati, in questo modo si possono individuare e correggere errori introdotti nella trasmissione.

QR Code

Le operazioni appena descritte, eseguite in tutti i sistemi di trasmissione digitale, trovano riscontro anche nei processi, precedentemente analizzati, di codifica e decodifica dei QR Code.

Nello specifico si possono considerare i dati da codificare nel QR Code come l'analogo di un flusso di informazione emesso da una sorgente digitale. Tali dati compaiono sotto forma di caratteri di testo, quindi simboli discreti, che devono essere rappresentati con una sequenza binaria, detta parola di codice (codeword). Questo mapping deve essere fatto nel modo più efficiente possibile, per ridurre il numero di bit con cui rappresentare l'informazione, e senza perdite (lossless), per restituire i dati originali nel processo inverso di decodifica. In questo consiste la codifica di sorgente nei QR Code. Ci sono quattro diverse modalità di codifica utilizzabili, in base al tipo di caratteri contenuti nella stringa dei dati: la modalità numerica, alfanumerica, byte e Kanji. Ciascuna modalità usa un metodo diverso per convertire i simboli in stringhe di bit ed è progettata per creare stringhe di lunghezza minore possibile per i particolari caratteri usati. Più precisamente, nelle specifiche dei QR Code, con data codeword si fa

riferimento a sotto-stringhe di 8 bit della sequenza di bit ottenuta dalla codifica dei dati di input, alla cui estremità iniziale sono presenti gli indicatori della modalità di codifica e del contatore di caratteri.

Per quanto riguarda la codifica di canale, nei QR Code sono adottati i codici Reed-Solomon per la rilevazione e la correzione di errori. A partire dai blocchi di data codeword, vengono calcolati i simboli per correggere gli errori e concatenati alle data codeword. Si può pensare a un blocco codificatore che riceve in ingresso le sequenze delle data codeword e le associa deterministicamente a sequenze più lunghe, le parole di codice, creando una struttura di ridondanza utile in fase di decodifica per la correzione degli errori.

Infine, anche nei QR Code si può trovare la modulazione e il concetto di canale. Nella fase di modulazione ciascun bit della sequenza dei dati codificati viene convertito singolarmente da informazione digitale a informazione analogica, ovvero mappato nel corrispondente simbolo che consiste in quadrato bianco (bit 0) o nero (bit 1), detto modulo. La traduzione dei bit in moduli chiari o scuri è sicuramente un tipo di modulazione molto semplice ma efficace, può essere complicata verosimilmente aggiungendo altri piani colore o altre forme geometriche, ottenendo di conseguenza un alfabeto di simboli più esteso. In aggiunta, in questo passo si tiene conto dell'applicazione del pattern di mascheramento su determinati moduli della matrice con lo scopo di aiutare lo scanner nella fase di demodulazione, evitando errori nell'identificazione dei quadrati con i dati.

A livello fisico, interviene un canale di comunicazione rumoroso che fornisce alla destinazione una versione alterata del QR Code. Questo a causa delle distorsioni che vengono introdotte sui simboli (cioè i moduli): il danneggiamento parziale o totale, le condizioni luminose che ne alterano il colore, l'orientazione anomala che ne altera la forma, il sensore RGB della fotocamera che non capta direttamente il colore nero e bianco ma li ottiene rispettivamente dall'assenza di colore e della combinazione di tutti i colori, e tante altre tipologie di distorsione.

La destinazione riceve la versione distorta del QR Code, diversa da quella presente in ingresso al canale, e opera una demodulazione per riottenere la stringa di bit dei dati. Il demodulatore prende una decisione sul simbolo (modulo) ricevuto, afflitto da rumore, decidendo così quale simbolo (modulo) è stato originariamente tramesso, secondo un determinato criterio. Una volta ottenuto il simbolo, ovvero il modulo, si recupera il bit associato. Sicuramente più la modulazione è complessa, ovvero maggiore è il numero di simboli trasmissibili (costellazione), più il processo di decisione è complesso e soggetto a errori.

Capitolo 3

Standard ISO/IEC 18004:2015 [1]

I QR Code, insieme ai relativi processi di codifica e decodifica, sono regolati da due importanti standard.

- JIS (Japanese Industrial Standards) X 0510 pubblicato nel gennaio 1999, sostituito prima da JIS X 0510:2004 (20-11-2004) e attualmente da JIS X 0510:2004 (22-01-2018). Questa ultima pubblicazione ha lo stesso contenuto dello standard ISO/IEC 18004:2015.
- ISO/IEC 18004 (QR Code modello 1 e 2) pubblicato dalla International Organization for Standardization nel giugno 2000, sostituito prima da ISO/IEC 18004:2006 (QR Code 2005) e attualmente da ISO/IEC 18004:2015 (QR Code).

L'attuale standard ISO/IEC 18004:2015 definisce nel dettaglio le caratteristiche dei QR Code, i metodi di codifica e decodifica dei dati, le versioni, le dimensioni, il processo per la correzione degli errori, altri requisiti e parametri tecnici.

Esistono numerose versioni di QR Code, ciascuna identificata da due parametri:

1. il numero della versione (V), da 1 a 40
2. il livello di correzione degli errori (E), distinto in 4 categorie intermedie L, M, Q e H

3.1 Caratteristiche

Il QR Code è un codice bidimensionale composto da una matrice quadrata contenente moduli quadrati. La rappresentazione dei dati avviene nel seguente modo:

- il modulo quadrato scuro (nero) corrisponde al bit 1
- il modulo quadrato chiaro (bianco) corrisponde al bit 0

Figura 3.1: QR Code versione 1-L

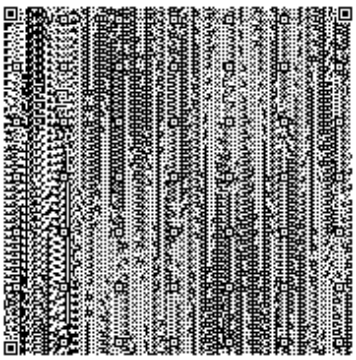


Figura 3.2: QR Code versione 40-L

3.1.1 Versioni

La dimensione del QR Code dipende dalla versione. Escludendo la quiet zone, la versione 1 (Fig. 3.1) presenta una matrice di 21 x 21 moduli, fino ad arrivare alla versione 40 (Fig. 3.2) con una matrice di 177 x 177 moduli. In ciascuna versione avviene un incremento di 4 x 4 moduli rispetto alla precedente.

3.1.2 Caratteri codificabili

Il QR Code codifica le seguenti tipologie di caratteri:

- caratteri numerici: numeri 0 - 9
- caratteri alfanumerici: numeri 0 - 9, lettere maiuscole A - Z, nove caratteri speciali (spazio, $ % * + - . / :)
- caratteri di 1 byte: ISO/IEC 8859-1 (e.g. lettere minuscole)
- caratteri Kanji: JIS X 0208

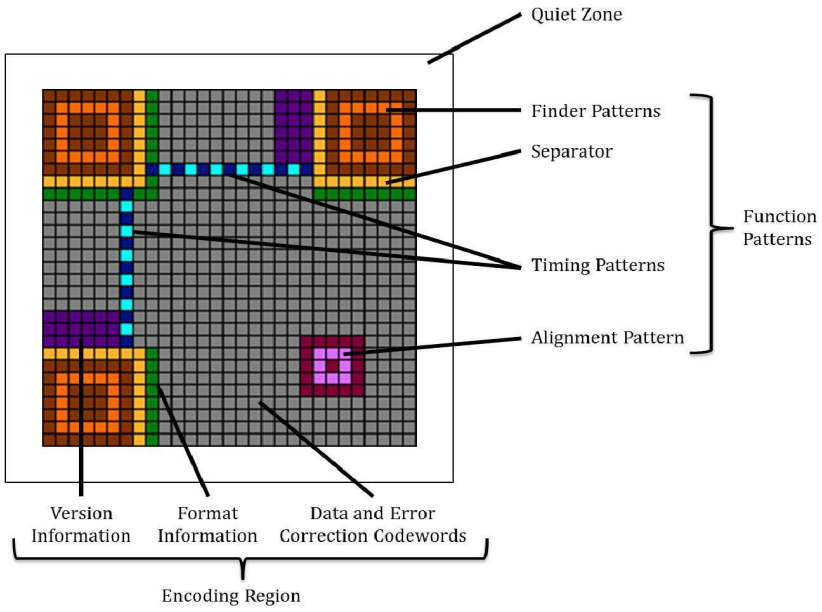


Figura 3.3: Struttura di un QR Code [12]

3.1.3 Livello di correzione degli errori

Ci sono quattro livelli di correzione degli errori attraverso i codici di Reed-Solomon, che permettono al massimo di recuperare una percentuale (approssimativa) diversa delle data codeword nel QR Code:

- Low (L): 7%
- Medium (M): 15%
- Quartile (Q): 25%
- High (H): 30%

All'aumentare del livello di correzione degli errori, diminuisce la capacità di immagazzinamento dei dati, a parità di versione.

3.1.4 Capacità massima

La versione 40 di un QR Code con livello di correzione degli errori L rappresenta la versione più capacitiva dei QR Code. Essa può contenere al massimo:

- dati numerici: 7089 caratteri
- dati alfanumerici: 4296 caratteri
- dati di 1 byte: 2953 caratteri

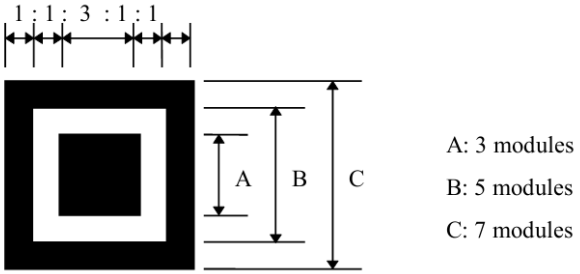


Figura 3.4: Finder Pattern [12]

- dati Kanji: 1817 caratteri

3.2 Struttura

Ciascun QR Code è costituito da moduli quadrati disposti su una matrice quadrata, contiene la regione di codifica e i pattern funzionali. Il simbolo è circondato su tutti e quattro i lati da una zona di bordo, detta quiet zone. La struttura è illustrata in Fig. 3.3.

3.2.1 Pattern funzionali

I pattern funzionali sono usati per fornire allo scanner l'esatta posizione e la forma del QR Code. In particolare, essi comprendono i seguenti elementi.

- Finder pattern

  Tre identici finder pattern sono localizzati nell'angolo in alto a sinistra, in alto a destra e in basso a sinistra del Qr Code. Essi sono costituiti da tre quadrati concentrici sovrapposti: rispettivamente, dal più esterno, un quadrato 7 x 7 di moduli scuri, un quadrato 5 x 5 di moduli chiari e, il più interno, un quadrato 3 x 3 di moduli scuri. Il rapporto tra le ampiezza dei moduli all'interno del pattern risulta essere 1:1:3:1:1 (Fig. 3.4).

  Grazie a questa progettazione, i finder pattern hanno una bassa probabilità di apparire in qualche altra parte del QR Code, e permettono una rapida identificazione del codice da parte del lettore. Essi hanno lo scopo di definire univocamente la posizione e l'orientamento del simbolo nello spazio.

- Separatore

  Il separatore è costituito da soli moduli chiari, ha lo spessore di un modulo e si trova tra ciascun finder pattern e la regione di codifica.

Livello di correzione degli errori	Indicatore
L	01
M	00
Q	11
H	10

Tabella 3.1: Indicatore del livello di correzione degli errori

- Timing pattern

  Sono presenti due timing pattern, uno verticale e uno orizzontale, entrambi costituiti da un'alternanza di moduli chiari e scuri, che deve iniziare e terminare con un modulo scuro. Nei QR Code il timing pattern orizzontale si trova nella sesta riga a partire dall'alto, tra i separatori dei finder pattern nel lato superiore, analogamente il timing pattern verticale è posizionato nella sesta colonna a partire da sinistra, tra i separatori dei finder pattern nel lato sinistro.

  Essi hanno lo scopo di aiutare a determinare la densità e la versione del simbolo, ma anche le posizioni dei moduli.

- Alignment pattern

  Gli alignment pattern sono presenti solamente nei QR Codes di versione 2 o superiore. Essi sono costituiti da tre quadrati concentrici sovrapposti: rispettivamente, dal più esterno, un quadrato 5 x 5 di moduli scuri, un quadrato 3 x 3 di moduli chiari e un quadrato centrale costituito da un singolo modulo scuro. Il numero e la posizione degli alignment pattern dipende dalla versione del simbolo.

  Essi hanno lo scopo di correggere la distorsione del simbolo quando è curvo.

3.2.2 Regione di codifica

La regione di codifica contiene i dati codificati e le informazioni relative al processo di codifica. Questa area è composta dalle seguenti parti:

- Informazione sul formato

  L'informazione sul formato è una sequenza di 15 bit contenente, a partire dal bit più significativo, 5 bit di dati con 10 bit per la correzione d'errore calcolati usando il codice BCH (15, 5). Dal bit più significativo, i primi 2 bit di dati indicano il livello di correzione degli errori del QR Code (Tab. 3.1), i successivi 3 bit indicano il pattern di mascheramento applicato (cfr. Sezione 3.3.4). Dopo aver calcolato i bit per la correzione degli errori, tutta

la stringa di 15 bit subisce un'operazione di XOR con la maschera 101010000010010, per assicurare che nessuna combinazione delle due informazioni codificate possa fornire una stringa di soli zeri.

I 15 bit vengono mappati nei moduli adiacenti al separatore del finder pattern superiore sinistro, disposti dall'alto al basso e poi da destra a sinistra (saltando il timing pattern), iniziando dal bit più significativo. In aggiunta, i bit vengono ripetuti negli otto moduli adiacenti al separatore sottostante il finder patter superiore sinistro e nei sette moduli adiacenti al separatore destro del finder pattern inferiore sinistro. Come riportato in Fig. 3.3. L'informazione sul formato viene ripetuta due volte nel simbolo per fornire ridondanza, dato che è fondamentale per la decodifica del QR Code.

- Informazione sulla versione

  L'informazione sulla versione è presente nei QR Code di versione 7 o superiore. Essa consiste in una sequenza di 18 bit contenente, a partire dal bit più significativo, 6 bit di dati (necessari per rappresentare 40 valori), con 12 bit per la correzione d'errore calcolati usando il codice di Golay (18, 6). I 6 bit indicano la versione del QR Code, da 1 a 40. Non viene applicata alcuna operazione di XOR con una maschera, dato che nessuna versione può essere identificata da una stringa di soli zeri.

  I 18 bit vengono mappati in un blocco 6 x 3 di moduli, sopra il timing pattern orizzontale e a destra del finder pattern superiore destro, e in un secondo blocco 3 x 6, a sinistra del timing pattern verticale e sopra il finder pattern inferiore destro, come riportato in Fig. 3.3. In entrambi i blocchi si dispone il bit più significativo nell'angolo superiore sinistro, procedendo da sinistra a destra e sulle righe inferiori. L'informazione sulla versione viene ripetuta due volte nel simbolo per fornire ridondanza, dato che è fondamentale per la decodifica del QR Code.

- Area con le data codeword e i simboli per la correzione degli errori

  Esclusi tutti i pattern funzionali, l'informazione sul formato e sulla versione, tutti i moduli rimanenti sulla matrice corrispondono ai bit delle parole di codice, con le data codeword e i simboli di parità. La disposizione dei bit segue le regole illustrate nel capitolo precedente.

3.2.3 Quiet Zone

Si tratta di una regione che delimita il QR Code su tutti e quattro i lati, ampia 4 moduli. Deve essere vuota, non contenere alcun tipo di elemento, e possedere lo stesso colore dei moduli chiari.

Essa ha lo scopo di evitare interferenze, durante la lettura del QR Code, dovute a testo o simboli nello spazio circostante.

3.3 Codifica

3.3.1 Modalità di codifica

Nella fase iniziale devono essere codificati i caratteri contenuti nella stringa dei dati in base al tipo, tra quelli supportati dai QR Code: numerico, alfanumerico, byte o Kanji. Per ciascun tipo di dato, esiste la corrispondente modalità di codifica. La diverse modalità utilizzabili, da quella numerica a quella Kanji, richiedono progressivamente più bit per rappresentare un carattere. Di conseguenza è conveniente adottare la modalità più efficiente, dopo aver analizzato i dati di input.

A volte può capitare che i dati contengano una parte di caratteri adatta per una certa modalità di codifica, e l'altra parte adatta per una diversa modalità: ad esempio una sequenza di numeri (modalità numerica) seguita da sequenze di lettere minuscole dell'alfabeto anglosassone (modalità byte). É possibile utilizzare in fase di codifica due modalità diverse. In questo modo sicuramente la stringa ottenuta ha la minor lunghezza in bit possibile, visto che si usa la modalità più efficace per i diversi tipi di caratteri. D'altro canto però, si deve valutare l'overhead che si genera a ogni cambiamento di modalità, dato che si deve specificare l'indicatore di modalità di codifica e l'indicatore del contatore di caratteri. Quindi non è detto che questo approccio sia il migliore per diminuire la lunghezza dei dati codificati, specialmente nel caso di pochi caratteri.

Ora si analizza in dettaglio la procedura da seguire in ciascuna modalità di codifica.

Modalità numerica

Nella modalità numerica la stringa dei dati da codificare viene suddivisa in gruppi di tre numeri e ciascun gruppo viene convertito nel corrispondente numero binario di 10 bit. Se il numero di cifre iniziali non è multiplo di tre, ci sono due casi: l'ultimo gruppo consta di soli due numeri e deve essere convertito nel relativo numero binario di 7 bit, oppure l'ultimo gruppo è composto da solo un numero e deve essere essere convertito nel numero binario di 4 bit equivalente. Se

Modalità	Indicatore
Numerica	0001
Alfanumerica	0010
Byte	0100
Kanji	1000

Tabella 3.2: Indicatore della modalità di codifica

V	Modalità numerica	Modalità alfanumerica	Modalità byte	Modalità Kanji
1-9	10	9	8	8
10-26	12	11	16	10
27-40	14	13	16	12

Tabella 3.3: Lunghezza dell'indicatore del contatore di caratteri per versione *V* e modalità di codifica

necessario, aggiungere zeri di padding nelle posizioni più significative. In questo modo non vengono sprecati bit, i bit richiesti sono il minimo indispensabile per rappresentare numeri decimali di tre, due e una cifra rispettivamente.

Infine la stringa codificata contiene, concatenate in ordine, tutte le stringhe di bit ottenute dai gruppi.

Modalità alfanumerica

Nella modalità alfanumerica la stringa dei dati da codificare viene suddivisa in gruppi di due caratteri. Per ciascun coppia si associa a entrambi i caratteri il valore corrispondente, da 0 a 44, secondo una tabella riportata nello standard. Il valore del primo carattere viene moltiplicato per 45 e gli viene sommato il valore del secondo carattere, poi il numero ottenuto viene convertito in un numero binario di 11 bit. Se il numero di caratteri iniziali non è multiplo di due, il valore del singolo carattere finale viene codificato con un numero binario di 6 bit. Se necessario, aggiungere zeri di padding nelle posizioni più significative.

Infine la stringa codificata contiene, concatenate in ordine, tutte le stringhe di bit ottenute dai gruppi.

Modalità byte

Nella modalità byte tutti i caratteri nella stringa dei dati da codificare vengono, singolarmente, mappati in un valore da 0 a 255. Si fa riferimento, per effettuare questa associazione, al gruppo di caratteri definito nello standard ISO 8859-1, i quali sono rappresentati da 8 bit e corrispondono al primo blocco di caratteri Unicode. Se necessario, aggiungere zeri di padding nelle posizioni più significative.

Condizione	Identificatore
$(r+c)$ mod 2 == 0	000
(r) mod 2 == 0	001
(c) mod 3 == 0	010
$(r+c)$ mod 3 == 0	011
($\lfloor r$ / 2$\rfloor$ + $\lfloor c$ / 3$\rfloor$) mod 2 == 0	100
($(r*c)$ mod 2) + ($(r*c)$ mod 3) == 0	101
(($(r*c)$ mod 2) + ($(r*c)$ mod 3)) mod 2 == 0	110
(($(r+c)$ mod 2) + ($(r*c)$ mod 3)) mod 2 == 0	111

Tabella 3.4: Condizioni e identificatori dei pattern di mascheramento: r e c indicano rispettivamente l'indice di riga e di colonna nella matrice, dove la posizione (0, 0) è fissata sul primo modulo in alto a sinistra

Infine la stringa codificata contiene, concatenate in ordine, tutte le stringhe di bit ottenute dai singoli caratteri.

Modalità Kanji

Nella modalità Kanji tutti i caratteri nella stringa dei dati da codificare vengono, singolarmente, mappati in un valore di due byte, in formato esadecimale, secondo la codifica Shift-JIS (Standard JIS X 0208). Successivamente tale valore viene compattato in una sequenza di 13 bit.

In particolare, questa modalità permette di codificare solo i caratteri Shift JIS corrispondenti a un valore esadecimale nel range 0x8140-0x9FFC e 0xE040-0xEBBF. Per i caratteri nel primo intervallo, il primo passo consiste nel sottrarre 0x8140 al valore esadecimale. Poi si divide il risultato nei due byte che lo compongono, quello più significativo e quello meno significativo. Si moltiplica il byte più significativo per 0xC0 e lo si somma al byte meno significativo. Infine, si converte il risultato nel corrispondente numero binario di 13 bit. Se necessario, aggiungere zeri di padding nelle posizioni più significative. Per i caratteri nel secondo intervallo la procedura è sostanzialmente la stessa, con la sola differenza che nel primo passo, anziché sottrarre 0x8140 al valore esadecimale, va sottratto 0xC140.

Alla fine la stringa codificata contiene, concatenate in ordine, tutte le stringhe di bit ottenute dai singoli caratteri.

3.3.2 Indicatore della modalità e del contatore di caratteri

La stringa dei dati codificati, affinché in fase di decodifica si risalga alla modalità di codifica usata e al numero di caratteri su cui è stata applicata, deve essere fatta precedere, nell'ordine, dall'indicatore della modalità di codifica e dall'indicatore del contatore di caratteri. In questo modo si ottiene un segmento.

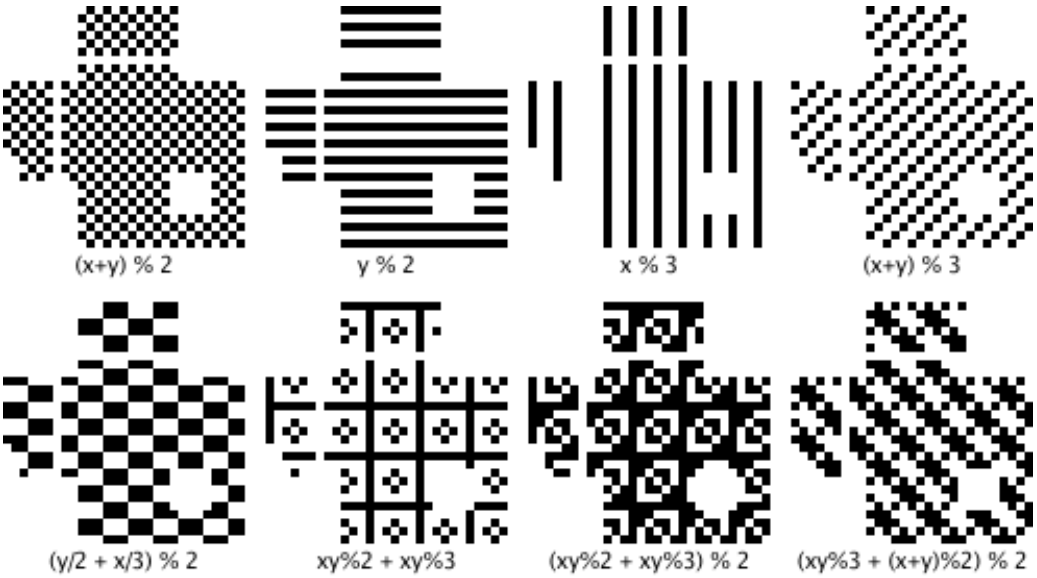


Figura 3.5: Gli otto pattern di mascheramento e le condizioni associate (Alexey Tikhonov. *8 possible XOR-masks*. Feb. 2019. URL: https://www.researchgate.net/publication/331165555_On_Double-Sided_QR-Codes (visitato il 08/09/2022))

L'indicatore della modalità di codifica consta di 4 bit, e viene scelto in base alla modalità, come in Tab. 3.2. Si può notare che si tratta di un codice a prefisso di lunghezza fissa.

L'indicatore del contatore di caratteri indica il numero di bit di dati codificati all'interno di un segmento. Esso ha una lunghezza in bit variabile in base alla modalità di codifica usata e alla versione del QR Code, come mostrato in Tab. 3.3.

3.3.3 Padding nella generazione delle data codeword

La generazione della data codeword avviene a partire dai dati codificati, cioè la sequenza di bit ottenuta da un segmento o da più segmenti concatenati, più l'eventuale terminatore. Nello specifico avviene una suddivisione in gruppi di 8 bit, dove ciascun ottetto rappresenta la singola data codeword.

Tuttavia, se la lunghezza della stringa di bit non è tale da terminare completamente una data codeword di 8 bit, ovvero non è multipla di 8, si devono aggiungere dei bit a 0 di padding. Se il numero data codeword non è sufficiente per soddisfare la capacità del QR Code, per quella determinata versione e quel livello di correzione degli errori, si devono aggiungere come padding le seguenti data codeword ripetute in alternanza: 11101100 e 00010001.

3.3.4 Pattern di mascheramento

Nel disporre i bit nei moduli della matrice spesso si creano combinazioni che possono potenzialmente ostacolare il processo di decodifica: troppi moduli dello stesso colore vicini, sequenze di moduli simili a pattern funzionali come il finder pattern o i timing pattern, etc. Per questo motivo ai dati viene applicato un pattern di mascheramento. Esso consiste formalmente in un'operazione di XOR tra i bit dei dati mappati nei moduli della matrice e un egual numero di bit di una matrice di tutti zeri (maschera), tranne specifici bit a 1, la cui posizione soddisfa una determinata condizione. Alla fine si rimappa il risultato ottenuto nei moduli. Operativamente l'operazione consiste nell'invertire il colore dei moduli che si trovano in precise posizioni.

Esistono otto tipi di pattern di mascheramento possibili: ciascuno è caratterizzato dalla condizione che indica le posizioni dei bit a 1 nella maschera, e dal proprio identificatore binario, da indicare nell'informazione del formato. Nello specifico le condizioni sono riportate in Tab. 3.4 e i pattern di mascheramento sono rappresentati in Fig. 3.5

Per capire quale pattern di mascheramento applicare al QR Code vanno valutati tutti singolarmente: per ciascun pattern si calcola il valore totale di penalità secondo quattro regole.

Ciascuna delle seguenti condizioni comporta una penalità p:

- ogni gruppo di 5 o più moduli dello stesso colore in una riga o colonna $\Rightarrow p = 3+n$, dove n rappresenta il numero di moduli aggiuntivi rispetto ai primi cinque;
- ogni gruppo 2 x 2 di moduli dello stesso colore nella matrice $\Rightarrow p = 3$;
- una disposizione orizzontale o verticale di moduli corrispondente al finder pattern, affiancata su uno o su entrambi i lati da 4 moduli chiari $\Rightarrow p = 40$;
- una maggiore quantità di moduli scuri rispetto a quelli chiari, o viceversa, quindi maggiore è lo sbilanciamento e più tale penalità è grossa $\Rightarrow p = 10\lfloor\left(|100\frac{\#moduliscuri}{\#modulitotali} - 50|\right)/5\rfloor$.

Viene applicato il pattern di mascheramento che ha ottenuto la penalità più piccola.

3.4 Correzione degli errori

3.4.1 Codici Reed-Solomon

I QR Code utilizzano i codici di Reed-Solomon[1] per la rilevazione e la correzione degli errori. Questi appartengono alla famiglia dei codici a blocco lineari, più precisamente sono codici

[1] Inventati da Irving S. Reed e Gustave Solomon nel 1960 al *Lincoln laboratory* (RS) del Massachusetts Institute of Technology (MIT)

BCH (Bose-Chauduri-Hocquenghem) non binari, la cui cardinalità dell'alfabeto dei simboli tipicamente è 2^m, con $m \geq 3$ [9].

Data un alfabeto finito $\mathscr{A}$ di q simboli, in un codice a blocco le parole composte da k simboli di dati sono mappate dal codificatore in parole di n simboli, dette parole di codice, terminate da n- k simboli di parità la correzione degli errori. Un codice è in grado di riconoscere quali simboli di una parola sono stati corrotti al canale e correggerli grazie alla ridondanza aggiunta. Infatti delle q^n parole di codice, solo q^k sono parole valide. Più le parole di codice differiscono tra di loro, ovvero maggiore è la ridondanza aggiunta, minore è la probabilità che vengano confuse in ricezione a causa degli errori introdotti dal canale.

In un codice a blocco lineare le parole formano un sottospazio vettoriale dello spazio composto da vettori di n-elementi appartenenti al campo finito, o campo di Galois, $GF(q)$. In questi codici, per descrivere quanto differiscono due parole, è definita la distanza di Hamming tra due parole **a** e **b** di codice di n simboli come:

$$d_H(\mathbf{a},\mathbf{b}) = \sum_{i=0}^{n} y_i \text{ dove } y_i = 1 \text{ se } a_i \neq b_i,\ y_i = 0 \text{ altrimenti} \tag{3.1}$$

La minimima distanza di Hamming di un codice è $d_{min} = min\{d_H(\mathbf{x},\mathbf{y})\}$, dove **x** e **y** sono due parole di codice diverse fra di loro.

La distanza minima di Hamming dei codici RS è:

$$d_{min} = n - k + 1 \tag{3.2}$$

Quindi i RS sono ottimi perché raggiungono la massima distanza minima di Hamming possibile per un codice lineare a blocco (n,k), ovvero soddisfano superiormente il bound di Singleton $d_{min} \leq n - k + 1$.

Il numero massimo di errori sui simboli che un codice può rilevare sempre è $d_{min} - 1$, mentre il numero massimo di errori che può correggere sempre è $\lfloor \frac{d_{min}-1}{2} \rfloor$. Quindi, per (3.2), la capacità correttiva dei codici RS è massima, ed è legata alla quantità di ridondanza aggiunta al blocco, ovvero $n-k$: si possono correggere fino a $\lfloor \frac{n-k}{2} \rfloor$ errori.

3.4.2 Generazione dei simboli per la correzione degli errori

Attraverso i codici RS, vengono generati i simboli per la correzione degli errori, che poi sono aggiunti alla sequenza delle data codeword. Come detto nella sezione 3.1, ci sono quattro livelli di correzione degli errori, ciascuno offre una capacità di recupero dei dati diversa: più il livello

di correzione è elevato e maggiore è il numero di simboli di parità aggiunti. I simboli di parità possono correggere le data codeword afflitte da errori o erasure: la differenza sta nel fatto che gli ultimi sono errori che avvengono in una posizione conosciuta, diversamente dagli errori che avvengono in posizioni non note a priori. Un erasure è un simbolo non scansionabile o non decodificabile, un errore è un simbolo decodificato erroneamente.

Indicando con e è il numero di erasure, t è il numero di errori e d è il numero di simboli per la correzione degli errori, si ha la seguente formula che fornisce il numero di erasure e di errori correggibili dai codici di Reed-Solomon:

$$e + 2t \leq d \tag{3.3}$$

Specifiche

Lo standard specifica l'utilizzo bit-a-bit dell'aritmetica modulo 2 e byte-a-byte dell'aritmetica modulo 100011101 per il calcolo dei simboli per la correzione degli errori.

Questo significa utilizzare i codici Reed-Solomon su un campo finito, o campo di Galois, con 256 elementi $GF(2^8)$ e polinomio primitivo $1 + x^2 + x^3 + x^4 + x^8$. I simboli appartenenti a tale campo sono tutti rappresentabili attraverso 8 bit, quindi numeri decimali nell'intervallo tra 0 e 255.

Qualsiasi elemento in $GF(256)$ può essere rappresentato come una potenza di 2, 2^n con $0 \leq n \leq 255$. Questo è triviale per gli elementi in {1, 2, 4, 8, 16, 32, 64, 128}, ma è possibile anche per tutti gli altri elementi poiché i numeri maggiori di 255 subiscono un'operazione di XOR con 285, il corrispondente numero decimale di 100011101 (come definito dallo standard), così da ricadere ciclicamente negli elementi rappresentabili in $GF(256)$. Ad esempio, in questo GF, 29 corrisponde a 2^8 perché 256 (2^8), maggiore di 255, subisce l'operazione 256 XOR $285 = 29$.

In questo modo è possibile compiere le operazioni di moltiplicazione e divisione più facilmente, riconducendosi a ad addizioni e sottrazioni degli esponenti delle potenze di 2. Per compiere le associazioni tra numeri maggiori di 255 e 2^n ($0 \leq n \leq 255$), nella pratica, si memorizzano delle tabelle ricavate a partire dal polinomio primitivo e dall'elemento primitivo del campo finito.

Le operazioni di addizione e sottrazione nel campo di Galois sono la stessa operazione: infatti una volta compiuta normalmente l'addizione o la sottrazione tra due elementi, si applica l'operazione (bit-a-bit) di modulo 2 (come definito dallo standard), ma questo coincide con

l'applicare all'inizio l'operazione di XOR bit-a-bit. Nel campo di Galois il negativo di un numero positivo coincide con la rappresentazione del corrispondente numero positivo, poiché la somma, equivalente all'operazione di XOR, tra un numero e se stesso da come risultato una stringa di zeri, ovvero l'elemento neutro. Le operazioni di moltiplicazione e divisione sono semplificate dalla rappresentazione dei numeri come potenze di 2: si ricorre ai logaritmi in base 2 e si trasforma il prodotto ab in $2^{\log_2 a+\log_2 b}$, analogamente per il rapporto, ma con la differenza all'esponente. Nel caso in cui il valore risultante dell'esponente sia maggiore di 255, si effettua l'operazione di modulo 255.

Procedura

Innanzitutto la sequenza con tutte le data codeword deve essere divisa nel numero di blocchi richiesto dalla versione e dal relativo livello di correzione degli errori. Quindi lo standard specifica, per ogni blocco, il numero totale di simboli dei dati e di correzione degli errori (data codeword e simboli di parità) e il numero di simboli dei dati (data codeword) presenti, cioè i parametri (n,k) che caratterizzano i codici RS. Per generare i simboli di correzione degli errori di ciascun blocco si osservano i passi seguenti.

1. Creare il polinomio del messaggio $m(x)$, cioè il polinomio che ha come coefficienti le data codeword del blocco, dove la prima data codeword d_{k-1} è il coefficiente del termine di grado più alto e l'ultima data codeword d_0 è il coefficiente del termine di grado più basso.
$$m(x) = d_{k-1}x^{k-1} + d_{k-2}x^{k-2} + ... + d_0 \tag{3.4}$$

2. Creare il polinomio generatore $g(x)$, cioè il polinomio ottenuto dal prodotto seguente, dove α è l'elemento primitivo su $GF(2^8)$ e n è il grado del polinomio generatore, corrispondente al numero di simboli di parità da ottenere.
$$g(x) = (x-\alpha^0)(x-\alpha^1)...(x-\alpha^{n-1}) \tag{3.5}$$

3. Eseguire la divisione (lunga) tra il polinomio del messaggio $m(x)$ e il polinomio generatore $g(x)$, prima modificando il polinomio del messaggio $m(x)$ nel seguente modo, per assicurarsi che l'esponente del termine più elevato non diventi troppo piccolo durante la divisione, indicato con k il numero di data codeword nel blocco.
$$m(x) = m(x)x^k \tag{3.6}$$

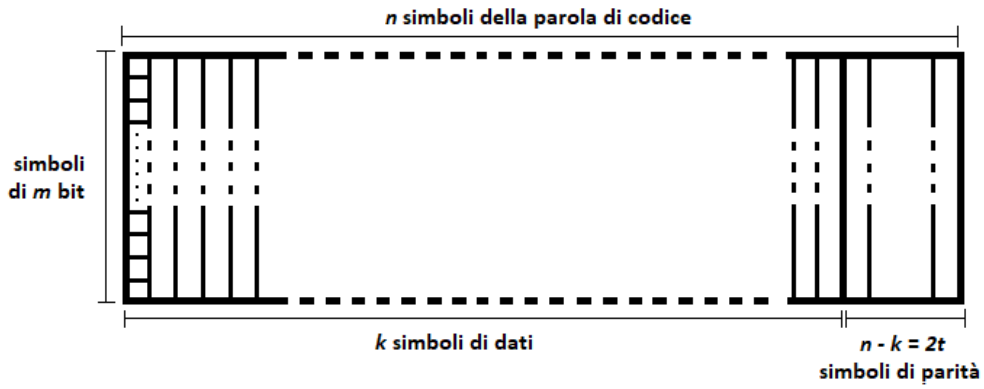


Figura 3.6: Struttura della parola di codice di Reed-Solomon

Dalla divisione si ottengono i polinomi quoziente $q(x)$ e resto $r(x)$ legati da questa relazione:

$$m(x) = g(x)q(x) + r(x) \tag{3.7}$$

I passi richiesti nella divisione sono pari al numero di data codeword.

In questa fase interviene l'aritmetica del campo di Galois vista in precedenza: l'operazione di moltiplicazione avviene sfruttando la somma tra logaritmi, con l'eventuale operazione di modulo per l'esponente, e nel passo in cui normalmente, durante la divisione tra polinomi, viene eseguita la sottrazione tra il risultato ottenuto dalla moltiplicazione e il polinomio del messaggio o il resto, si effettua l'analoga operazione (nel GF) di XOR.

4. Considerare i coefficienti del polinomio di resto $r(x)$ ottenuto in (3.7) come i simboli per la correzione degli errori del blocco, dove il coefficiente del termine di grado più alto è il primo simboli di parità e_{n-k-1} e il coefficiente del termine di grado più basso è l'ultimo simbolo di parità del blocco e_0.

$$r(x) = e_{n-k-1}x^{n-k-1} + e_{n-k-2}x^{n-k-2} + \ldots + e_0 \tag{3.8}$$

Al termine dell'operazione si può identificare in ciascuna coppia, composta dal blocco di k data codeword e il relativo blocco di ridondanza con gli n - k simboli di parità, la parola di codice di Reed-Solomon formata da n simboli di m bit (in questo caso 8 bit), come mostrato in Fig. 3.6.

3.5 Decodifica

3.5.1 Dimensione del modulo

All'inizio della decodifica del QR Code, dopo aver determinato la soglia adeguata per stabilire il colore chiaro o scuro dei moduli, si cerca di determinare un finder pattern sulla base delle caratteristiche descritte nella Sezione 3.2.1. Appena viene trovata una zona che le soddisfa, si determina la larghezza orizzontale e le coordinate del centro dell'elemento. Si ripete la procedura anche per gli altri due finder pattern. Se non viene trovato alcun candidato, si deve correggere la soglia impostata inizialmente.

Analizzando le coordinate dei centri dei finder pattern e determinando qual è il finder pattern superiore sinistro, è possibile stabilire l'orientazione del QR Code.

Indicando con B la larghezza orizzontale del finder pattern superiore sinistro e con C la larghezza orizzontale del finder pattern superiore destro, la dimensione nominale del modulo X è definita dalla seguente formula:

$$X = (B+C)/14 \tag{3.9}$$

3.5.2 Versione del QR Code

Per definire la versione del QR Code è necessario calcolare la distanza tra i centri dei due finder pattern nel lato superiore. Indicata con A questa misura e con X la dimensione nominale del modulo precedentemente trovata in (3.9), il numero della versione V viene provvisoriamente calcolato attraverso la seguente formula:

$$V = (A/X - 10)/4 \tag{3.10}$$

Se il risultato ottenuto in (3.10) è uguale o inferiore a 6, allora questa è definitivamente la versione del QR Code. Altrimenti va trovata e decodificata l'informazione sulla versione, presente solo nelle versioni superiori alla 6.

Capitolo 4

Aumento della capacità dei QR Code

La capacità di immagazzinamento dei QR Code tradizionali, con moduli bianchi e neri, raggiunge il valore massimo nella versione 40-L di 4296 caratteri alfanumerici. Tuttavia, nonostante la considerevole capacità, si stanno studiando ugualmente degli approcci per aumentare la capacità dei QR Code per i seguenti motivi: aumento delle informazioni mantenendo la compattezza del simbolo, maggiore robustezza e infine la riduzione dello spazio occupato sul materiale su cui avviene la stampa. Grazie alla flessibilità nella struttura di questi simboli, esistono diverse possibilità per estendere i limiti di capacità dei dati immagazzinati.

In letteratura ci si è focalizzati principalmente sulle seguenti tecniche:

- *Colored QR Code* (CQR Code)
- Per-Colorant-Channel Color Barcode
- Multiplexing di QR Code
- Compressione dei dati

4.1 Colored QR Code (CQR Code)

I Colored QR Code, abbreviati in CQR Code, sono stati proposti da Vizcarra et al. in [15] e [14]. Questi simboli condividono la stessa struttura dei QR Code con moduli bianchi e neri: contengono i pattern funzionali e la regione di codifica. Oltre all'uso dei colori, l'ulteriore differenza consiste nel fatto che non sono usati gli alignment e i timing pattern tra i pattern funzionali: si tralasciano poiché la dimensione è ridotta, quindi meno soggetta a deformazioni, e non deve essere determinata, dato che è fissa.

Coppia di bit	Colore del modulo
00	Rosso
01	Verde
10	Blu
11	Bianco

Tabella 4.1: Mappatura tra la coppia di bit e il colore del modulo [15]

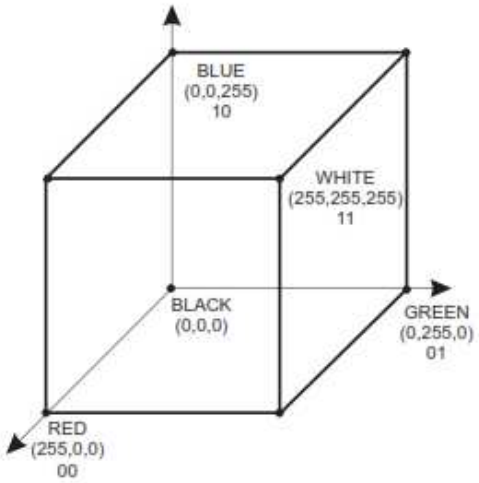


Figura 4.1: Spazio colore RGB e la distanza tra i colori [15]

La principale innovazione riguarda l'utilizzo di diversi colori nella rappresentazione dei moduli. In particolare nella fase di codifica, durante la mappatura tra i bit dei dati codificati e i moduli nella matrice, si associano gruppi di più bit a un modulo che può assumere numerosi colori predeterminati, i singoli bit non corrispondono più semplicemente a un modulo chiaro o scuro. Quindi si adotta una modulazione più complessa.

In base al numero di colori utilizzati si possono distinguere due diversi modelli di CQR Code: i CQR Code-5 e i CQR Code-9.

4.1.1 CQR Code-5

I CQR Code-5 [15] sono composti da una matrice quadrata di 49 x 49 moduli (cfr. versione 8 dei QR Code standard), dove la dimensione di ogni modulo è *n x n* pixel, in base alla dimensione reale del simbolo. In totale sono disponibili 2401 moduli, dei quali 2209 sono usati per contenere i dati codificati.

I CQR Code-5 sfruttano 5 colori per rappresentare le informazioni: il rosso, il verde, il blu, il nero e il bianco. In particolare i colori rosso, verde, blu e bianco sono utilizzati per i moduli nella regione di codifica, ovvero quelli contenenti le informazioni codificate, mentre il

Figura 4.2: Esempio di CQR Code-5 [15]

colore nero (insieme al bianco) è usato solo per i moduli necessari per l'allineamento in fase di decodifica, cioè i pattern funzionali.

Come si vedrà più nel dettaglio nella codifica, 2 bit di informazione sono associati a un modulo che può assumere 4 colori, quindi i 2209 moduli contenuti rappresenteranno 4418 bit: in particolare 1024 bit di dati e 3392 bit di parità (2 bit, corrispondenti a 1 modulo, non sono utilizzati).

Codifica

Il processo di codifica dei CQR Code-5 rispecchia quasi interamente i passi visti nella Sezione 3.1. I punti di scostamento sono principalmente due: il primo riguarda la fase di generazione delle data codeword e conseguentemente dei simboli di parità, il secondo è legato alla modulazione della stringa binaria dei dati nei simboli, cioè i moduli dei 4 colori citati in precedenza.

Innanzitutto, dopo aver ottenuto la stringa di bit attraverso l'opportuna modalità di codifica dei i dati iniziali, le data codeword hanno una lunghezza di 16 bit, anziché gli 8 bit dei QR Code tradizionali. Il blocco dei simboli di parità è ottenuto dal blocco di data codeword, attraverso i codici RS. I Reed-Solomon considerano un blocco di k data codeword e aggiungono un blocco di $n-k$ simboli di parità, per formare una parola di codice di n simboli. Più nello specifico vengono usati i $RS(n,k)$ con $n = 276$ e $k = 64$, e con polinomio primitivo per la generazione dei simboli di parità $x^{16} + x^{12} + x^3 + 1$. Le parole di codice hanno struttura, dove d indica le data codeword ed e i simboli per la correzione degli errori

$$RS(276,64) = [d_1 ... d_{64} e_1 ... e_{212}] \tag{4.1}$$

Con queste caratteristiche, i RS possono correggere fino a $t = \frac{n-k}{2} = 106$ simboli errati nelle

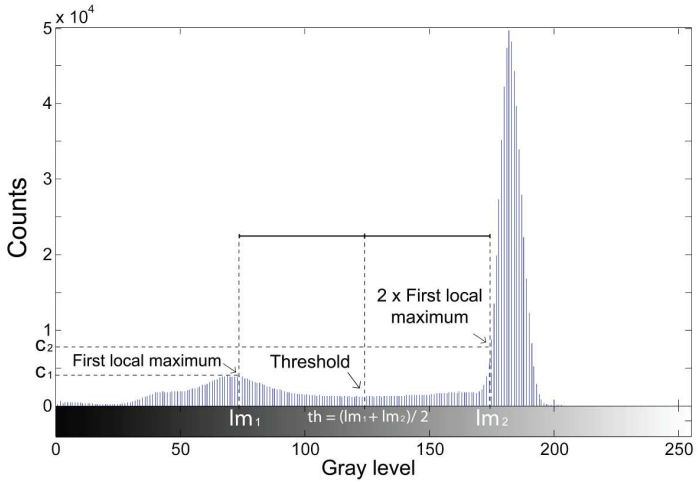


Figura 4.3: Istogramma tipico per calcolare la soglia *th* [15]

parole di codice, che corrispondono al 38.41% dei simboli totali.

Poi i simboli dei dati e per la correzione degli errori vengono combinati insieme durante la fase di interleaving, nello stesso ordine descritto nella Sezione 2.1.

Per quanto riguarda la generazione dei moduli, si comincia dalla suddivisione in gruppi di due bit della stringa binaria ottenuta dall'interleaving. Ciascun gruppo viene mappato in un modulo di un colore scelto tra quattro possibili colori dello spazio colore RGB di 24 bit, dove ogni piano colore è rappresentato con 8 bit. I colori del modulo associabile alla coppia di bit sono il rosso, il verde, il blu e il bianco, seguendo il mapping mostrato in Tab. 4.1. Sono stati scelti proprio questi colori poiché la loro distanza sullo spazio colore RGB è massima, come riportato in Fig. 4.1. In questo modo viene semplificata l'operazione di thresholding durante la fase di decodifica. Si procede con la disposizione dei moduli, rappresentati dall'opportuno colore, nella matrice seguendo lo stesso ordine usato nei QR Code tradizionali, ma senza applicare alla fine alcun pattern di mascheramento.

Un esempio di CQR Code-5 ottenuto dalla codifica è riportato in Fig. 4.2.

Decodifica

Il processo di decodifica inizia dall'acquisizione di una foto del CQR Code attraverso una fotocamera digitale.

L'identificazione del simbolo prevede la ricerca dei tre finder pattern, per stabilire la posizione del CQR Code e correggere l'angolo di rotazione. Per localizzare i finder pattern si deve prima convertire l'immagine a colori in scala di grigi e poi determinare la soglia th per il thresholding,

$$th = (lm_1 + lm_2)/2 \tag{4.2}$$

Algorithm 1 COLOR-THRESHOLD(C, th) [15]

```
for i = 1 to C.height do
    for j = 1 to C.width do
        MaxC = max(C(i, j, :))                    ▷ massimo valore RGB per il pixel
        MinC = min(C(i, j, :))                    ▷ minimo valore RGB per il pixel
        D = MaxC − MinC
        if D > th then
            if MaxC == C(i, j, R) then                                  ▷ rosso
                C(i, j, RGB) = {255, 0, 0}
            else if MaxC == C(i, j, G) then                             ▷ verde
                C(i, j, RGB) = {0, 255, 0}
            else                                                        ▷ blu
                C(i, j, RGB) = {0, 0, 255}
            end if
        else
            if C(i, j, RGB) < {th, th, th} then                         ▷ nero
                C(i, j, RGB) = {0, 0, 0}
            else                                                        ▷ bianco
                C(i, j, RGB) = {255, 255, 255}
            end if
        end if
    end for
end for
return C
```

dove lm_1 è il livello di grigio del primo massimo locale c_1 nell'istogramma dell'immagine (Fig. 4.3) e lm_2 è il più piccolo livello di grigio corrispondente a un valore $c_2 = 2c_1$ [15]. Le variabili c_1 e c_2 rappresentano i contatori rispettivamente dei livelli di grigio lm_1 e lm_2.

Una volta determinata la soglia th, si applica l'Alg. 1 di thresholding per recuperare dall'immagine acquisita C del CQR Code i colori originali dei moduli, descritti da opportuni valori di combinazione dei piani colore nello spazio RGB. In questo modo si ottiene una matrice di moduli con i colori stabiliti in fase di codifica, in cui i finder pattern sono ben definiti. Quindi si può procede utilizzando lo stesso algoritmo di localizzazione riportato nello standard ISO/IEC 18004, continuando poi con la decodifica, durante la quale si riapplica la mappatura inversa tra moduli e bit, si correggono gli eventuali errori e si riottengono le data codeword.

Vantaggi

Il CQR Code-5 ha il principale vantaggio di incrementare la capacità di immagazzinamento rispetto al QR Code con moduli bianchi e neri. Ogni modulo colorato non corrisponde più a un singolo bit, ma a una coppia di bit. Di conseguenza la capacità all'incirca raddoppia (disallineamento dovuto all'assenza di alcuni pattern funzionali), traducendosi nell'immagazzinamento

CQR Code	Correttamente decodificati	Errore medio percentuale sui simboli
1	15	0.97
2	15	0.53
3	14	5.49
4	15	8.79
5	15	8.24
6	13	14.8
7	15	7.25
8	15	3.04

Tabella 4.2: Risultati della decodifica degli 8 CQR Code-5 [15]

Figura 4.4: Foto per ciascun CQR Code-5 con la più alta percentuale d'errore sui simboli. Nella prima e seconda riga ci sono rispettivamente i CQR Code dal numero 1 a 4 e da 5 a 8, disposti da sinistra verso destra [15]

del doppio dei bit di informazione rispetto al classico QR Code: 4418 bit di dati codificati (inclusi i simboli di parità) in 2209 moduli.

L'altro vantaggio riguarda il riutilizzo dei già esistenti algoritmi di localizzazione e correzione dell'orientamento del QR Code, descritto dallo standard ISO/IEC 18004:2015, applicati all'immagine acquisita del CQR Code e convertita in scala di grigi.

Svantaggi

La presenza di una modulazione leggermente più complessa tra l'informazione binaria e i simboli sulla matrice, che sfrutta 5 colori anziché solo il bianco e il nero, da un lato rappresenta un vantaggio in termini capacitivi, ma dall'altro comporta più errori in fase di demodulazione poiché i simboli tra cui compiere una decisione, sulla base di quelli rumorosi ricevuti, sono più numerosi e aumenta la sensibilità alle distorsioni introdotte dal canale. In particolare, le distorsioni sui colori avvengono in fase di stampa, a causa della qualità della stampante, oppure in fase di acquisizione del CQR Code, per via della tipologia di fotocamera o per le condizioni

d'illuminazione dell'ambiente.

Per quanto riguarda la fotocamera, un sensore che è dotato di una bassa risoluzione o che, verosimilmente, si trova a lavorare in condizioni luminose inadeguate, può introdurre distorsioni sui colori. Inoltre, anche gli stessi sensori che catturano i canali nello spazio RGB comportano delle differenze nella rilevazione di un stesso colore, che dipendono da produttore a produttore in base alle specifiche di fabbricazione.

Per analizzare quanto si può sbagliare durante la decodifica di questi QR Code a 5 colori, in [15] viene riportato un test compiuto su 15 foto di 8 diversi CQR Code-5, le cui dimensioni di stampa sono *1.3 cm x 1.3 c*m, che quindi forniscono una capacità di trasmissione di circa *605 bit/cm*^{2}. Le immagini sono state acquisite con una fotocamera di 3.2 Megapixel di uno smartphone, sotto diverse condizioni di illuminazione. Nella Tab. 4.2 sono riportati i risultati ottenuti dal test.

Dai risultati si può dedurre che tutte le foto dei CQR Code-5 numero 1, 2, 4, 5, 7 e 8 sono state correttamente decodificate, mentre per il CQR Code-5 numero 3 una foto non è stata decodificata e per il CQR Code-5 numero 6 addirittura due foto non sono state correttamente decodificate. Molto probabilmente, in questi casi, le importanti distorsioni introdotte sui colori non hanno permesso di recuperare il contenuto originale. Tuttavia, in generale l'errore medio sui simboli dei moduli distorti è abbastanza contenuto, raggiunge un picco di 14.8%, ma risulta sempre inferiore alla massima capacità correttiva dei RS per questi CQR Code-5, cioè 38.41%. Quindi si può affermare che quasi la totalità dei CQR Code-5 scansionati è stata correttamente decodificata. La Fig. 4.4 riporta, per ciascun CQR Code analizzato, le foto che ha ricevuto la più alta percentuale d'errore sui simboli .

4.1.2 CQR Code-9

I CQR Code-9 hanno la stessa struttura dei CQR Code-5: sono composti da una matrice quadrata di 49 x 49 moduli e sono disponibili 2401 moduli, dei quali 2209 sono usati per contenere i dati codificati.

I QR Code-9 sfruttano 9 colori per rappresentare le informazioni: il rosso, il verde, il blu, il ciano, il magenta, il giallo, il grigio, il nero e il bianco. In particolare tutti gli 8 colori, eccetto il nero, sono utilizzati per i moduli nella regione di codifica, ovvero quelli contenenti le informazioni codificate, mentre il colore nero (insieme al bianco) è usato solo per i moduli necessari per l'allineamento in fase di decodifica, cioè i pattern funzionali.

Tripletta di bit	Colore del modulo
000	Rosso
001	Verde
010	Blu
011	Ciano
100	Magenta
101	Giallo
110	Bianco
111	Grigio

Tabella 4.3: Mappatura tra la tripletta di bit e il colore del modulo [14]

Figura 4.5: Esempio di CQR Code-9 [14]

Come si vedrà più nel dettaglio nella codifica, 3 bit di informazione sono associati a un modulo che può assumere 8 colori, quindi i 2209 moduli contenuti rappresenteranno 6627 bit: in particolare 2048 bit di dati e 4576 bit di parità (3 bit, corrispondenti a 1 modulo, non sono utilizzati).

Codifica

Il processo di codifica è per la maggior parte coincidente con quello dei CQR Code-5 e presenta delle differenze solo nelle specifiche dei codici di Reed-Solomon e nella mappatura tra l'informazione binaria e i colori dei moduli, visto che i colori possibili sono 8 e non 4.

In particolare vengono usati i $RS(n,k)$ con $n = 414$ e $k = 128$, e con polinomio primitivo per la generazione dei simboli di parità $x^{16} + x^{12} + x^3 + 1$, lo stesso adottato nei CQR Code-5. Le parole di codice hanno la forma mostrata in (4.3), indicate con d le data codeword (sempre di 16 bit) ed e i simboli per la correzione degli errori.

$$RS(414,128) = [d_1 ... d_{128} e_1 ... e_{286}] \tag{4.3}$$

Con queste caratteristiche, i RS possono correggere fino a $t = \frac{n-k}{2} = 143$ simboli errati nelle

parole di codice, che corrispondono al 34.54% dei simboli totali.

Per quanto riguarda la creazione dei moduli, si comincia dalla suddivisione in gruppi di tre bit della stringa binaria ottenuta dall'interleaving tra le data codeword e i simboli di parità. Come mostrato in Tab. 4.3, ciascuna tripletta binaria viene mappata in un modulo di un colore scelto tra otto possibili colori dello spazio colore RGB di 24 bit, dove ogni piano colore è rappresentato con 8 bit. Si procede con la disposizione dei moduli, rappresentati dall'opportuno colore, nella matrice seguendo lo stesso ordine usato nei QR Code tradizionali, ma senza applicare alla fine alcun pattern di mascheramento.

Un esempio di CQR Code-9 ottenuto dalla codifica è riportato in Fig. 4.5.

Decodifica

I CQR Code-9 utilizzano lo stesso processo di decodifica dei CQR Code-5. Tuttavia, poiché i colori usati per rappresentare i moduli sono più numerosi, l'algoritmo di decisione dei colori è più articolato.

Il nuovo algoritmo decisionale di segmentazione [14] ha come obiettivo la determinazione del colore predominante in ogni pixel della foto acquisita. Presi in input l'immagine C del CQR Code e la soglia th precedentemente calcolata, con lo stesso procedimento visto nei CQR Code-5, l'algoritmo determina per ogni pixel il massimo, il medio e il minimo valore nello spazio colore RGB, per trovare poi la distanza dai nove possibili colori (rosso, verde, blu, ciano, magenta, giallo, grigio, nero e bianco). L'algoritmo dettagliato è mostrato in Alg. 2.

Vantaggi

Il CQR Code-9 ha il principale vantaggio di incrementare la capacità di immagazzinamento rispetto al QR Code con moduli bianchi e neri. Ogni modulo colorato non corrisponde più a un singolo bit, ma a una tripletta di bit. Di conseguenza l'aumento della capacità consiste in un fattore 3, che si traduce nel poter contenere all'interno del simbolo approssimativamente (disallineamento dovuto all'assenza di alcuni pattern funzionali) il triplo dei bit di informazione rispetto al classico QR Code: 6627 bit di dati codificati (inclusi i simboli di parità) in 2209 moduli.

Allo stesso modo del CQR Code-5, l'altro vantaggio riguarda il riutilizzo dei già esistenti algoritmi di localizzazione e correzione dell'orientamento del QR Code, descritto dallo standard ISO/IEC 18004:2015, applicati all'immagine acquisita del CQR Code e convertita in scala di grigi.

Algorithm 2 CQR Code-9 Segmentation(C, th) [14]

for $i = 1$ to $C.height$ **do**
  for $j = 1$ to $C.width$ **do**
    $MaxC = max(C(i, j, :))$ ▷ valore RGB massimo per il pixel
    $MidC = mean(C(i, j, :))$ ▷ valore RGB medio per il pixel
    $MinC = min(C(i, j, :))$ ▷ valore RGB minimo per il pixel
    $D = MaxC - MinC$
    if $D < th/2$ **then**
      if $MinC > 1.5th$ **then**
        $C(i, j, RGB) = \{255, 255, 255\}$ ▷ bianco
      else if $MaxC < th$ **then**
        $C(i, j, RGB) = \{0, 0, 0\}$ ▷ nero
      else
        $C(i, j, RGB) = \{127, 127, 127\}$ ▷ grigio
      end if
    else
      if $MaxC == C(i, j, R)$ **then**
        if $MaxC - MidC < th/2$ and $MidC == C(i, j, G)$ **then**
          $C(i, j, RGB) = \{255, 255, 0\}$ ▷ giallo
        else if $MaxC - MidC < th/2$ and $MidC == C(i, j, B)$ **then**
          $C(i, j, RGB) = \{255, 0, 255\}$ ▷ magenta
        else
          $C(i, j, RGB) = \{255, 0, 0\}$ ▷ rosso
        end if
      else if $MaxC == C(i, j, G)$ **then**
        if $MaxC - MidC < th/2$ and $MidC == C(i, j, R)$ **then**
          $C(i, j, RGB) = \{255, 255, 0\}$ ▷ giallo
        else if $MaxC - MidC < th/2$ and $MidC == C(i, j, B)$ **then**
          $C(i, j, RGB) = \{0, 255, 255\}$ ▷ ciano
        else
          $C(i, j, RGB) = \{0, 255, 0\}$ ▷ verde
        end if
      else
        if $MaxC - MidC < th/2$ and $MidC == C(i, j, R)$ **then**
          $C(i, j, RGB) = \{255, 0, 255\}$ ▷ magenta
        else if $MaxC - MidC < th/2$ and $MidC == C(i, j, G)$ **then**
          $C(i, j, RGB) = \{0, 255, 255\}$ ▷ ciano
        else
          $C(i, j, RGB) = \{0, 0, 255\}$ ▷ blu
        end if
      end if
    end if
  end for
end for
return C

Distanza CQR Code-9 (*cm*)	Correttamente decodificati	Errore medio % sui simboli
6	0	100.00
7	10	18.28
8	10	19.34
9	10	14.54
10	10	23.38
11	10	29.83
12	10	32.89
13	8	33.71
14	0	39.13
15	0	41.30
16	0	42.10
17	0	61.10
18	0	86.81
19	0	87.22
20	0	90.77
21	0	100.00
22	0	100.00

Tabella 4.4: Risultati della decodifica a 17 diverse distanze dei CQR Code-9 [14]

Svantaggi

I fattori che limitano i CQR Code-9 e le problematiche connesse corrispondo a quelli analizzati nei CQR Code-5. nella Sezione 4.1.1.

Tuttavia, rispetto al CQR Code-5, il CQR Code-9 utilizza un numero di colori più elevato in fase di codifica, 9 contro 5, che con buona probabilità rende la decodifica più propensa a errori visto che in fase di demodulazione i simboli tra cui compiere una decisione sono maggiori, rendendo così il rumore più invasivo.

Nella Sezione 4.1.1, si è notato dai risultati del test eseguito sui CQR Code-5, che comunque l'errore medio compiuto sui simboli decodificati era abbastanza contenuto. Nonostante l'utilizzo di cinque colori, le distorsioni hanno compromesso lievemente la fase di decodifica, mantenendola corretta nella quasi totalità dei casi. Nel seguente paragrafo si analizzano i risultati del test condotto sui CQR Code-9.

In [14] viene proposto un test compiuto su 10 foto di diversi CQR Code-9, acquisite a 17 diverse distanze tra i 6 cm e i 22 cm. Le immagini sono state prodotte dalla stampante Ricoh MP C2051 e le dimensioni di stampa dei codici sono 1.3 cm x 1.3 cm. Le immagini sono state acquisite con una fotocamera di 16 Megapixel di uno smartphone Samsung Galaxy S5, sotto diverse condizioni di illuminazione. Nella Tab. 4.4 sono riportati i risultati ottenuti dal test.

Dai risultati si può dedurre che tutte le foto dei CQR Code-9 acquisite a una distanza compresa tra 7 cm e 13 cm sono state correttamente decodificate, eccetto solamente due su dieci nel

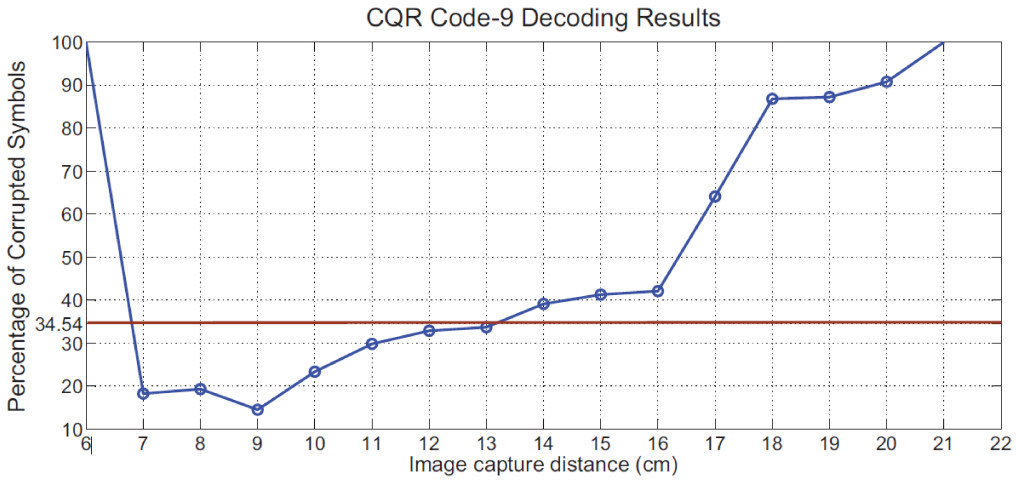


Figura 4.6: Grafico dell'errore percentuale medio sui simboli durante la decodifica di CQR Code-9 a diverse distanze [14]

caso di 13 cm. Invece, per distanze molto piccole, sotto i 7 cm, e distanze molto grandi, sopra i 13 cm, nessuna foto di CQR Code-9 è stata correttamente decodificata.

Molto probabilmente, nei casi in cui non è avvenuta la decodifica, la risoluzione e la qualità della fotocamera hanno condizionato fortemente il riconoscimento dei colori dei moduli: la messa a fuoco a tali distanze può essere risultata inadeguata e da molto lontano la qualità della foto acquisita può non essere stata sufficiente. Potevano essere ottenuti dei risultati migliori con delle fotocamere di qualità superiore. Inoltre, anche per quanto riguarda le foto correttamente decodificate, in generale l'errore medio sui simboli dei moduli distorti è abbastanza importante, come mostrato in Fig. 4.4. L'errore raggiunge un picco di 33.71% e sfiora, risultando comunque inferiore, la massima capacità correttiva dei RS per questi CQR Code-5, cioè 34.54%. Mentre nelle foto non decodificate, l'errore medio risulta sempre superiore alla percentuale massima di correzione.

Infine dal test si può affermare che più della metà delle foto dei CQR Code-9 non sono state decodificate correttamente, ben 102 su 170: l'adozione di nove colori nella modulazione aumenta di molto la capacità del simbolo, ma influisce gravemente sul corretto riconoscimento dei moduli da parte dello scanner.

4.1.3 Confronto tra CQR Code-5 e CQR Code-9

Le due versioni di CQR Code si differenziano sul numero di colori usati per rappresentare i moduli: i CQR Code-5 utilizzano 5 colori, mentre i CQR Code-9 utilizzano 9 colori. Entrambi possiedono una capacità di immagazzinamento dati superiore ai QR Code classici, con moduli bianchi e neri. In particolare i CQR Code-5 hanno una capacità doppia visto che i moduli

sono associati a 2 bit, mentre i CQR Code-9 hanno una capacità tripla dato che i moduli sono associati a 3 bit.

I CQR Code-9 sicuramente hanno capacità maggiore rispetto ai CQR Code-5, ma comportano una mappatura più complessa tra l'informazione binaria e i colori dei moduli. Questo si traduce in decisioni da parte della destinazione (scanner) più difficili e soggette a errori in fase di demodulazione, riflettendosi poi sulla decodifica. Dai test presentati nelle sezioni precedenti emerge chiaramente che i CQR Code-9 risultano difficilmente decodificabili, nella maggior parte dei casi la decodifica non avviene correttamente a causa delle numerose distorsioni sui colori introdotte dal canale, che superano la capacità correttiva dei codici RS. D'altro canto i CQR Code-5, pur avendo una capacità inferiore visto il minor numero di colori utilizzato, dal test effettuato risultano quasi sempre decodificabili correttamente, con una percentuale di errori dovuti alle distorsioni sui simboli rimediata dai codici RS. Questo si spiega considerando la maggiore limitatezza dei colori dei moduli e quindi un conseguente rumore che, mediamente, genera errori di decisione meno gravi lato decodificatore.

In conclusione, i CQR Code-5 offrono un ottimo compromesso in termini di capacità di immagazzinamento dei dati e di corretta decodifica.

4.2 Per-Colorant-Channel Color Barcode

Blasinski et al., in [2], propongono un nuovo modello per la costruzione di codici a barre bidimensionali a colori che permette di estendere i tradizionali codici monocromatatici a più colori. Vengono illustrate in dettaglio la fase di codifica e decodifica.

Codifica

Il processo di codifica è un'estensione diretta e triviale della metodologia usata per i QR Code standard. I dati vengono inizialmente suddivisi in tre gruppi indipendenti e ciascuno viene codificato nel relativo QR Code monocromatico, in bianco e nero, poi saranno tutti combinati insieme. La codifica del QR Code segue gli stessi passi visti nel Capitolo 2. Prima che avvenga la fusione, i tre QR Code vengono trasposti singolarmente nei colori ciano (C), magenta (M) e giallo (Y), ovvero i canali colore che solitamente vengono usati nella stampa. Ora i tre codici possono essere stampati in sovrapposizione per ottenere il QR Code a colori. Il processo viene illustrato in Fig. 4.7.

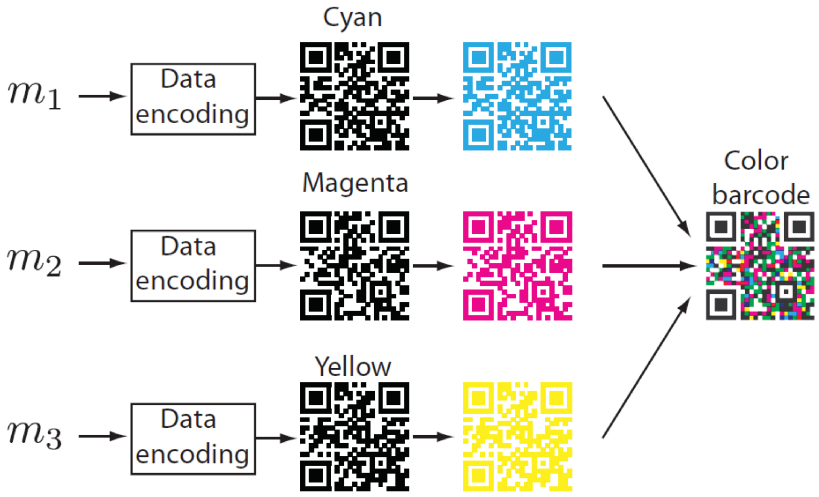


Figura 4.7: Codifica del QR Code attraverso la fusione di tre QR Code monocolore CMY (Per-Colorant-Channel Color) [2]

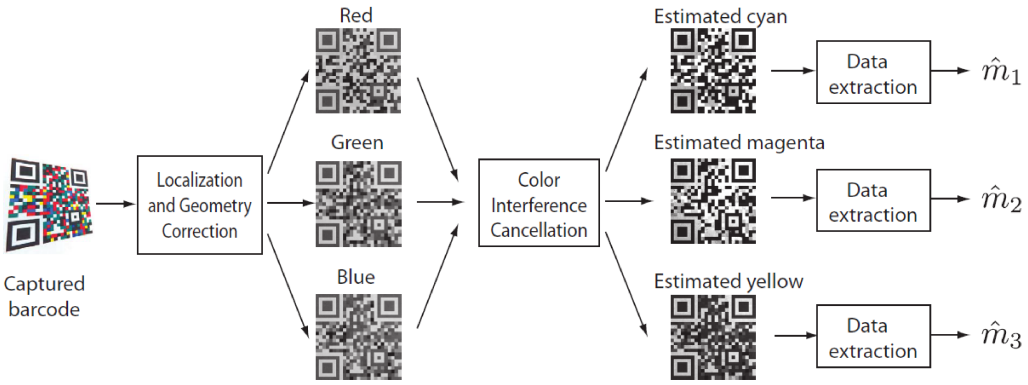
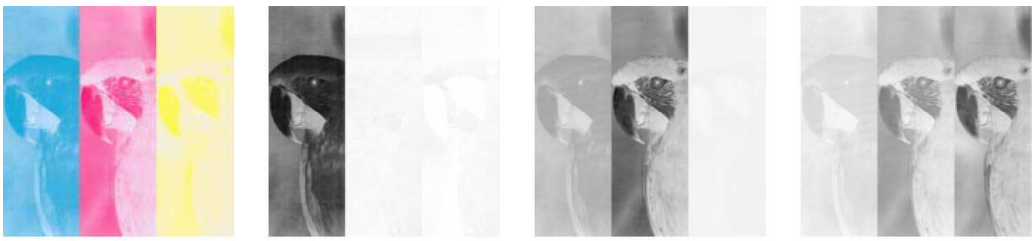


Figura 4.8: Decodifica del QR Code ottenuto dalla fusione di tre QR Code monocolore CMY (Per-Colorant-Channel Color) [2]

Decodifica

Il processo di decodifica è più complesso della codifica, dato che emergono delle problematiche dovute al recupero dei colori C, M e Y, usati in codifica, da parte del sensore della fotocamera.

Durante la decodifica si cerca di recuperare i dati immagazzinati nel QR Code analizzando, comunemente, un'immagine acquisita dalla fotocamera di uno smartphone. Prima viene localizzato il simbolo, convertendo l'immagine acquisita in scala di grigi e usando le tecniche dei QR Code monocromatici, e poi si scompone l'immagine nei tre piani colore dello spazio RGB, ovvero rosso (R), verde (G) e blu (B), catturati tipicamente dai sensori delle fotocamere digitali. In condizioni ideali, i colori C, M e Y possono essere ricavati a partire dai colori acquisiti tramite il sensore RGB della fotocamera, attraverso la seguente trasformazione, dove R, G e B sono normalizzati su 255:



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

Figura 4.9: Interferenze tra i canali colore C, M e Y della stampa e i canali colore R, G e B della fotocamera: (a) Stampa suddivisa in CMY (b) Scansione canale R (c) Scansione canale G (d) Scansione canale B [3]

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{4.4}$$

Dalla (4.4) emerge, ad esempio, che il ciano puro corrisponde a una rilevazione da parte della fotocamera di $R = 0, G = 1$ e $B = 1$, quindi assenza di rosso e presenza di verde e blu puri.

Tuttavia, nella realtà, la relazione espressa in (4.4) deve tenere in considerazione i disturbi che intervengono nelle mappatura tra i canali colore della stampa e della fotocamera (*cross channel coupling*): infatti i valori rilevati dal sensore della camera, per ciascun canale R, G e B, sono influenzati da rumore, come illustrato in Fig. 4.9, dovuto a diversi fattori. In particolare spiccano i tipi di pigmenti utilizzati dalla stampante, che fanno variare la purezza del ciano, magenta e giallo in fase di stampa, oppure la tipologia di costruzione del sensore stesso, o ancora le specifiche condizioni di illuminazione.

Esistono degli algoritmi di cancellazione delle interferenze sui colori che permettono di ricavare in modo esatto il valore dei canali C, M e Y dai canali R, G e B dell'immagine acquisita del QR Code. Tali algoritmi sono computazionalmente poco onerosi e forniscono un importante miglioramento delle prestazioni. Per applicarli è necessario determinare dei parametri per la cancellazione del rumore, che dipendono da vari fattori quali le condizioni luminose, il tipo di stampante e di fotocamera, attraverso due metodologie: un approccio che sfrutta dei blocchi di riferimento (*pilot block* PB) e un altro approccio iterativo che massimizza l'aspettazione (*expection maximization* EM) attraverso una funzione obiettivo. Il primo utilizza un aspetto particolare per rappresentare gli indicatori utilizzati nell'allineamento, ovvero i finder pattern, da cui stimare i parametri richiesti. Il secondo approccio stima iterativamente i parametri del modello e i canali colore della stampa, senza richiedere alcuna modifica nell'aspetto dei codici. [2]

In questo modo, dopo aver cancellato le interferenze, si ottengono tre immagini in scala

EC		Estimated Cyan					Estimated Magenta					Estimated Yellow					Monochrome
		Int. cancellation			AT	LT	Int. cancellation			AT	LT	Int. cancellation			AT	LT	
		PB + AT	PB + LT	EM			PB + AT	PB + LT	EM			PB + AT	PB + LT	EM			
L	SSR	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	85.19
	DSR	100	100	83.72	90.70	100	97.67	41.86	81.40	39.53	69.77	20.93	6.98	46.51	0	0	98.26
M	SSR	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	95.38
	DSR	100	100	85.71	92.86	100	92.86	38.10	88.10	52.38	40.48	45.24	26.19	54.76	0	0	100
Q	SSR	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	88.89
	DSR	97.67	100	88.37	95.35	100	93.02	34.88	81.40	55.81	58.14	30.23	44.19	58.14	0	0	100
H	SSR	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	95.56	87.41
	DSR	100	100	90.70	100	100	100	86.05	79.07	41.86	58.14	37.21	25.58	48.84	2.33	0	100
Total	SSR	95.00	95.00	95.00	95.00	95.00	95.00	95.00	95.00	95.00	95.00	95.00	95.00	95.00	95.00	95.00	89.16
	DSR	99.42	100	87.13	94.74	100	95.91	50.29	82.46	47.37	56.73	33.33	25.73	52.05	0.58	0	99.58

Tabella 4.5: Risultati del test effettuato sui QR Code a colori e monocromatici, utilizzando o meno gli algoritmi di cancellazione delle interferenze e con algoritmi di thresholding differenti, dove SSR rappresenta il rate di successo della sincronizzazione e DSR il rate di successo della decodifica [2]

di grigi corrispondenti ai tre livelli di colore C, M e Y. Da queste possono essere recuperati, sfruttando gli esistenti algoritmi di thresholding (fissi *FT* , adattivi *AT* o localizzati *LT*), i bit di informazione di ogni QR Code monocromatico e si procede singolarmente con i passi della decodifica standard per estrarre i dati contenuti. Questa avviene secondo la procedura vista nel Capitolo 2. Infine vengono riuniti i dati che erano stati codificati indipendentemente. Il processo viene illustrato in Fig. 4.8.

Vantaggi

Il principale vantaggio ottenuto dalla fusione di tre QR Code monocolore C, M e Y è l'incremento della capacità di immagazzinamento dei dati nel QR Code finale, che raggiunge il triplo della capacità dei QR Code tradizionali. A differenza degli altri QR Code che sfruttano i colori, come i CQR Code, i dati non vengono codificati insieme in moduli che portano più informazioni sotto forma di bit, ma in tre QR Code monocromatici indipendenti, poi convertiti nei canali colore C, M e Y per la stampa e sovrapposti: in questo modo è possibile utilizzare degli approcci per limitare le interferenze tra i canali RGB acquisiti dalla fotocamera e quelli di stampa, basandosi su dei modelli fisici, che migliorano le prestazioni della decodifica.

Inoltre, insieme alla cancellazione delle interferenze, la codifica indipendente sui tre canali colore C, M e Y favorisce una maggiore robustezza verso le distorsioni sul colore introdotte in fase di stampa. Infatti, nonostante nell'insieme i livelli possano risultare mal decodificati, questo metodo è più robusto poiché i dati sono codificati e decodificati indipendentemente da ciascun livello colore, contrariamente alle altre tecniche che codificano insieme tutti i dati basandosi su colori prodotti dalla combinazione di diversi piani colore. Queste ultime risentono

maggiormente delle distorsioni sul colore, dato che comportano un errore su una porzione più ampia di informazione, anziché in parallelo su una porzione ridotta. [2]

I test in [2] hanno considerato QR Code a colori del modello proposto e QR Code monocromatici in bianco e nero, con tutti i livelli di correzione possibili, dato che influenzano il processo di decodifica. Le 180 immagini dei QR Code colorati e le 540 immagini dei QR Code monocromatici sono state prodotte da una stampante HP Color LaserJet 4700 dn e acquisite da nove differenti smartphone. I risultati sono mostrati in Tab. 4.5. Il rate di successo della sincronizzazione (SSR) ottenuto dai QR Code a colori, ovvero la percentuale per cui il decodificatore ha localizzato correttamente il codice nell'immagine, risulta essere complessivamente molto elevato (95%), superiore anche al rate di sincronizzazione ottenuto dai QR Code monocromatici (90%). Per quanto riguarda il rate di successo della decodifica (DSR), nel canale del ciano si raggiungono valori molto elevati, tra il 90% e il 100%, indipendentemente dalla presenza o meno della cancellazione delle interferenze sul colore e dal tipo di soglia usata durante il thresholding. Mentre nel canale del magenta e soprattutto del giallo gli algoritmi di cancellazione delle interferenze provocano in generale un notevole innalzamento del rate di decodifica, visto che tipicamente sono i due canali più impattati dalle interferenze sul colore: nel magenta risulta migliore l'algoritmo basato sui i blocchi di riferimento (PB) e nel giallo l'algoritmo di massimizzazione dell'aspettazione (EM).

Infine il nuovo modello di QR Code proposto riutilizza metodi già esistenti: la stessa procedura di codifica e decodifica e gli stessi algoritmi di localizzazione e correzione dell'orientamento del QR Code monocolore, descritto dallo standard ISO/IEC 18004:2015.

Svantaggi

La modulazione usata in questi QR Code risulta abbastanza complessa, essa è stata studiata per essere resistente agli errori introdotti in fase di stampa e alla interferenze tra i diversi canali colore. Tuttavia, come si è visto anche nei CQR, la qualità della fotocamera e le condizioni luminose possono essere responsabili di errori durante la decodifica. Infatti non c'è alcun modo efficace per evitare le distorsioni sul colore causate alla base da una particolare tipologia di sensore della fotocamera , o da risoluzione non adeguata o da condizioni di luce estreme.

Un altro svantaggio legato al processo di codifica di questi QR Code è l'obbligo della stampa a colori, essi non possono essere riprodotti da stampanti che forniscono la stampa in bianco e nero. Tuttavia si tratta di una piccola limitazione dato che oggi la stampa a colori è diffusa, anche perché sono già molte le varianti esistenti a colori dei QR Code.

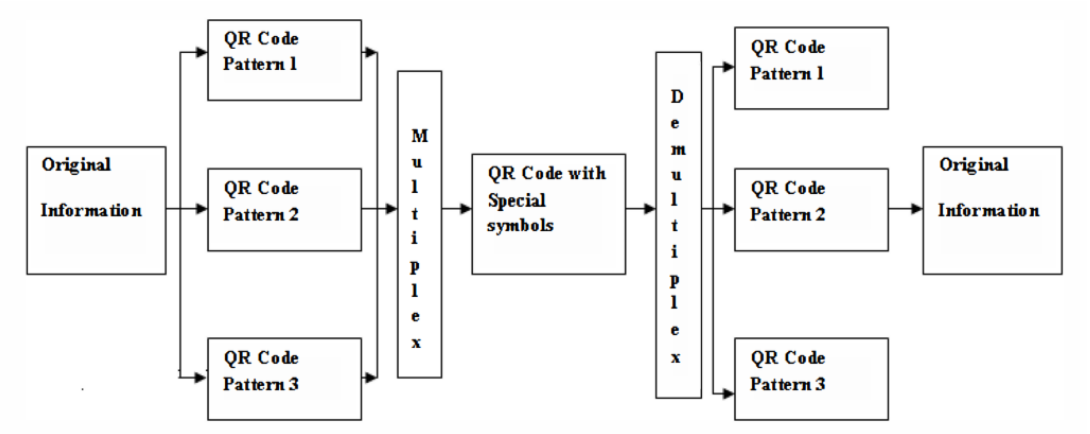


Figura 4.10: Panoramica del processo di codifica e decodifica riguardo il multiplexing dei QR Code [16]

Tripletta di bit	Simbolo speciale
000	\
001	/
010	v
011	^
100	>
101	<
110	⌞
111	⌝

Tabella 4.6: Esempio di mappatura tra le combinazioni di 3 bit e 8 simboli speciali durante il multiplexing [16]

Inoltre i QR Code proposti non possono essere letti da uno scanner di QR Code monocromatici, perciò richiedono obbligatoriamente uno specifico decodificatore di una certa complessità che esegua la procedura di decodifica analizzata precedentemente.

4.3 Multiplexing di QR Code

4.3.1 Multiplexing attraverso una codifica con simboli speciali

Vongpradhip e Sartid in [16] propongono una nuova tecnica per incrementare la capacità dei QR Code. Essa consiste nel fondere più QR Code codificati indipendentemente su diverse partizioni dei dati originali in un unico QR Code finale, che sfrutta simboli speciali bianchi e neri. Questo processo prende il nome di multiplexing.

In Fig. 4.10 viene illustrata una panoramica dei processi di codifica e di decodifica.

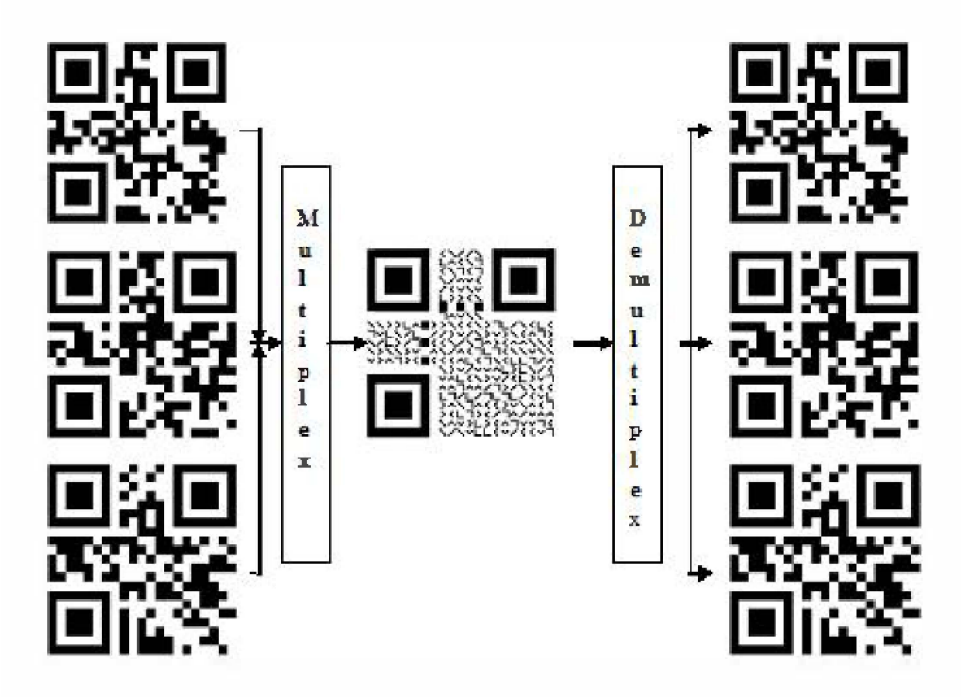


Figura 4.11: Esempio di QR Code ottenuto dal multiplexing di 3 QR Code [16]

Codifica

Il processo di codifica prevede una fase iniziale di suddivisione delle informazioni originali in n parti, ovvero sottostringhe di caratteri, che corrispondono al numero di QR Code da fondere in seguito. I dati in ogni parte sono poi codificati in modo indipendente nel corrispettivo QR Code monocromatico, secondo i passi visti nel Capitolo 2. Tutti i codici hanno la stessa versione.

A questo punto si procede con la fase di multiplexing degli n QR Code: tutti i moduli bianchi (bit 0) e neri (bit 1) corrispondenti alla stessa posizione nei codici vengono fusi sfruttando dei simboli speciali bianchi e neri, ad esclusione dei moduli dei finder pattern e dei timing pattern identici nelle n parti. A ciascuna combinazione di n bit ricavata da una precisa posizione viene associato, secondo la mappatura definita, un simbolo. Questi simboli rappresentano le informazioni codificate e assemblate, al posto dei tradizionali moduli quadrati, nel QR Code finale. L'alfabeto di tali simboli speciali deve avere una cardinalità pari a 2^n, dove n è il numero di QR Code in ingresso al multiplexing. Questo poiché ogni simbolo viene mappato su una combinazione di n bit (moduli) provenienti dalle n parti iniziali, quindi 2^n corrisponde proprio al numero di combinazioni possibili rappresentabili con n bit. Attualmente qualsiasi carattere sulla tastiera, pattern di caratteri speciali o simbolo speciale può essere utilizzato. In Tab. 4.6 è riportato un esempio di mappatura tra combinazioni di 3 bit, provenienti dai moduli di 3 QR Code, e $2^3 = 8$ simboli speciali.

Infine i simboli speciali, bianchi e neri, vengono posti sulla matrice insieme ai pattern

funzionali in bianco e nero.

Un esempio di QR Code ottenuto dal multiplexing di 3 QR Code e codificato sfruttando 8 simboli speciali è presente in Fig. 4.11.

Decodifica

Il processo di decodifica inizia dall'acquisizione di un'immagine del QR Code, che viene analizzata con gli esistenti algoritmi di localizzazione del codice, sfruttando i finder pattern e i timing patter. In seguito, attraverso un algoritmo di riconoscimento, vengono identificati i simboli speciali e si opera una mappatura inversa rispetto a quella usata in fase di codifica, riottenendo i corrispondenti bit di informazione di tutti i moduli delle n parti combinate insieme. Una volta elaborati i bit e recuperate le n stringhe binarie suddivise, si ricostruiscono gli n QR Code monocromatici mappando rispettivamente bit 1 e 0 in moduli neri e bianchi. Questa fase prende il nome di demultiplexing. [16]

A questo punto, riapplicando i passi di decodifica standard visti nel Capitolo 2, si ricavano i dati contenuti nei codici, che possono essere ad esempio scansionati da qualsiasi tradizionale lettore. Infine, i dati estratti da ogni QR Code sono concatenati insieme per ricomporre la stringa originale, suddivisa all'inzio della fase di codifica.

Vantaggi

Il vantaggio principale dovuto al processo di multiplexing è l'incremento della capacità del singolo QR Code: i dati iniziali vengono suddivisi in gruppi più piccoli di informazione, codificabili, di conseguenza, in QR Code più ridotti e compatti (meno capacitivi), che poi vengono fusi e rappresentati attraverso simboli speciali. In questo modo il QR Code ottenuto riesce a contenere, proporzionalmente al numero di codici coinvolti nel multiplexing, un multiplo dei dati immagazzinati nel QR Code tradizionale (con moduli bianchi e neri) con le analoghe dimensioni.

Inoltre, il processo per ottenere il nuovo modello di QR Code proposto utilizza, nei passi intermedi, le tecniche già esistenti per i QR Code standard riguardo la fase di codifica e decodifica e per gli algoritmi di localizzazione.

Svantaggi

Un importante svantaggio di questo approccio è legato all'utilizzo dei simboli speciali. Sicuramente essi richiedono un algoritmo di riconoscimento completamente diverso rispetto a quello

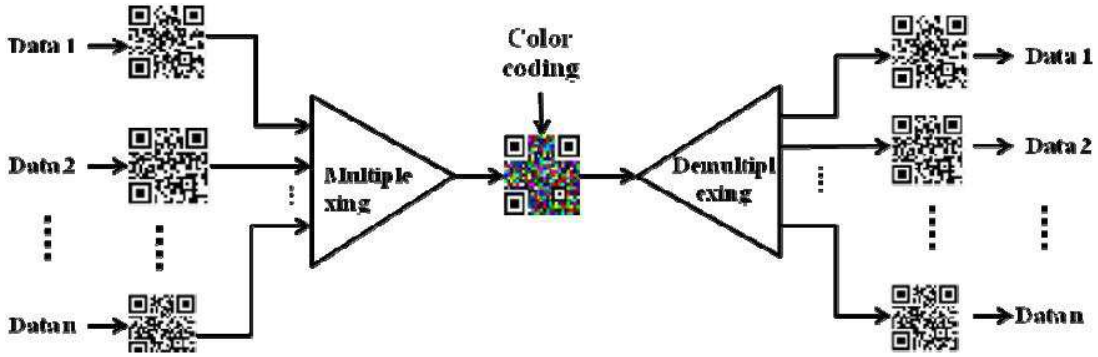


Figura 4.12: Panoramica del processo di codifica e decodifica riguardo il multiplexing, con la codifica a colori, dei QR Code [6]

usato per identificare i moduli bianchi e neri, o di altri colori, nei QR Code già esistenti. Quindi c'è un'importante riprogettazione che deve essere considerata per via del nuovo tipo di modulazione. Il decodificatore da architettare si basa in maniera importante sulla tipologia di simboli utilizzati, se è vero che da un lato può essere scelto un qualsiasi tipo di alfabeto, dall'altro si è vincolati ai soli simboli identificabili dall'algoritmo di riconoscimento usato.

In aggiunta, l'incremento di capacità del QR Code non è fisso, ma dipende dal numero di simboli che l'algoritmo è in grado di riconoscere: maggiore è il numero di simboli utilizzabili in codifica e maggiore è la quantità di QR Code che possono essere accorpati insieme nel codice finale.

4.3.2 Multiplexing attraverso una codifica con colori

In [6] Galiyawala et al. propongono una nuova tecnica che adotta il multiplexing di più QR Code della stessa versione per aumentare la capacità dei dati immagazzinati. Essa sfrutta una codifica con i colori nello spazio RGB, diversamente dai simboli speciali visti in precedenza, per rappresentare le diverse combinazioni di bit ottenute dalla fusione dei codici iniziali. Il risultato finale è un QR Code a colori che immagazzina più bit del corrispondente monocromatico.

In Fig. 4.12 viene illustrata una panoramica dei processi di codifica e di decodifica.

Codifica

Il processo di codifica inizia con la suddivisione dell'informazione originale in n partizioni, che vengono usate per generare individualmente i corrispettivi n QR Code, con moduli bianchi e neri e della stessa versione, secondo le tecniche di codifica esistenti. Questi QR Code sono poi destinati alla fase di multiplexing, in cui vengono fusi usando delle tecniche di codifica con i colori.

Bit QR Code 1	Bit QR Code 2	Bit QR Code 3	R	G	B	Colore
0	0	0	0	0	0	Nero
0	0	1	0	0	1	Blu
0	1	0	0	1	0	Verde
0	1	1	1	0	0	Rosso
1	0	0	0	1	1	Ciano
1	0	1	1	0	1	Magenta
1	1	0	1	1	0	Giallo
1	1	1	1	1	1	Bianco

Tabella 4.7: Esempio di mappatura tra le combinazioni di bit di 3 QR Code e 8 colori RGB durante il multiplexing [6]

Figura 4.13: Esempio di QR Code ottenuto dal multiplexing attraverso una codifica con colori di 3 QR Code [6]

In particolare, durante il multiplexing si considerano tutti i moduli bianchi e neri che si trovano nella stessa posizione in tutti i QR Code: il contenuto in parallelo di ciascuna posizione, identificato da una combinazione di n bit ottenuta dai moduli dei diversi codici, secondo cui il modulo scuro è un bit a 1 e il modulo chiaro è un bit a 0, viene rappresentato da un colore sullo spazio RGB, secondo la mappatura definita. Come per i simboli speciali, sono necessari 2^n colori distinti per rappresentare tutte le possibili combinazioni di n bit. Da questa procedura sono esclusi tutti i moduli corrispondenti alle posizioni dei pattern funzionali, dato che sono uguali in tutti i codici. In Tab. 4.7 è riportato un esempio di mappatura tra combinazioni di 3 bit, provenienti dai moduli di 3 QR Code, e $2^3 = 8$ colori nello spazio RGB.

Infine i moduli colorati vengono posti sulla matrice insieme ai pattern funzionali in bianco e nero.

Un esempio di QR Code ottenuto dal multiplexing di 3 QR Code e codificato sfruttando 8 colori nello spazio RGB è presente in Fig. 4.13.

Decodifica

Il processo di decodifica inizia dall'acquisizione di una foto del QR Code finale a colori attraverso una fotocamera digitale. L'identificazione del simbolo prevede la ricerca dei tre finder pattern, in modo tale da stabilire la posizione del codice e correggere l'angolo di rotazione. In seguito, attraverso opportuni algoritmi di thresholding, si recuperano dall'immagine i colori dei moduli. In particolare, separando i piani colore nello spazio RGB, si ottengono i valori di R, G e B di ciascun modulo del QR Code prodotto in fase di multiplexing. In questo modo si applica la mappatura inversa usata in codifica, riottenendo i corrispondenti gruppi di bit di informazione per ogni modulo rappresentante le n parti combinate insieme. Una volta elaborati i bit e recuperate le n stringhe binarie suddivise di ciascuna parte, si ricostruiscono gli n QR Code monocromatici mappando rispettivamente bit 1 e 0 in moduli neri e bianchi. Questa fase, speculare al multiplexing, prende il nome di demultiplexing. [6]

A questo punto i codici possono essere scansionati da qualsiasi lettore per ricavare i dati contenuti, applicando i passi standard di decodifica visti nel Capitolo 2. Infine, i dati estratti da ogni QR Code sono concatenati insieme per formare l'intera stringa originale, inizialmente suddivisa.

QR Code nel multiplexing	Colori distinti	Durata codifica (*s*)	Durata decodifica (*s*)
2	4	2.922	2.892
3	8	3.194	2.988
4	16	3.364	2.982
5	32	3.131	3.084
6	64	3.297	3.381
7	128	3.489	3.624
8	256	3.911	4.359
9	512	4.811	6.128
10	1024	6.229	11.406
11	2048	9.453	28.398
12	4096	15.787	91.393
13	8192	27.940	323.198
14	16384	53.157	1236.105

Tabella 4.8: Risultati del tempo di elaborazione in codifica e decodifica al variare dei QR Code intervenuti nel multiplexing [6]

Vantaggi

Il principale vantaggio del multiplexing di QR Code, attraverso una codifica con i colori, è l'incremento della capacità di immagazzinamento dei dati del QR Code ottenuto rispetto al corrispondente QR Code monocromatico. Questo aumento dipende da quanti colori distinti sono rappresentabili nello spazio RGB. Normalmente si utilizzano 8 bit per piano colore, quindi lo spazio RGB offre 2^8 x 2^8 x $2^8 = 2^{24}$, ovvero più di 16 milioni sfumature di colore diverse. La disponibilità di 2^{24} colori distinti permette in teoria di fornire in ingresso alla fase di multiplexing fino a un massimo di 24 QR Code monocromatici, che vengono fusi insieme dando origine a un singolo QR Code colorato con una capacità limite 24 volte superiore al QR Code standard. Alternativamente, una stessa quantità di dati può essere contenuta in un QR Code di dimensioni ridotte, in cui viene applicato il multiplexing, rispetto al corrispondente QR Code standard di dimensioni maggiori. Si può affermare che con questo metodo si superano considerevolmente tutti gli incrementi capacitivi portati dalle tecniche viste in precedenza.

Inoltre, il nuovo modello di QR Code proposto permette di riutilizzare nei passi intermedi di metodi già esistenti: la stessa procedura di codifica e decodifica e gli stessi algoritmi di localizzazione e correzione dell'orientamento dei QR Code monocromatici.

Svantaggi

L'incremento massimo della capacità prodotto da questo QR Code è veramente importante, tuttavia, dal punto di vista pratico emergono numerose limitazioni.

Sicuramente la problematica più grande riguarda il tempo richiesto dall'elaborazione durante la fase di decodifica, in particolare dal demultiplexing. Infatti, per un numero superiore a 10 QR Code su cui applicare il multiplexing, la durata del processo di decodifica supera i limiti temporali ragionevoli di attesa per una scansione veloce di un codice, la cui rapidità viene suggerita dallo stesso nome *Quick Response (QR) Code*.

In [6] viene eseguito un test che coinvolge il multiplexing e il demultiplexing da 2 a 14 QR Code di versione 2. Il processore che esegue le elaborazioni è un Intel Core i3-2310M 2.10 GHz. Nella Tab. 4.8 sono riportati tutti i risultati ottenuti. In generale si nota che il tempo richiesto dalla fase di codifica si mantiene abbastanza contenuto, al contrario il tempo per la decodifica esplode oltre il minuto superando i 12 QR Code combinati insieme. Con un numero minore di 9 QR Code la codifica e la decodifica impiegano tempi in media sui 3-4 *s*, da 11 o più si riscontra un aumento temporale vertiginoso che porta la decodifica fino a migliaia di secondi. Di conseguenza, per mantenere delle prestazioni in termini temporali accettabili, conviene eseguire il multiplexing limitandosi a un massimo di 10 QR Code.

Un altro problema che sorge in questi QR Code, ma che riguarda tutti i QR Code che fanno uso dei colori nel processo di codifica, è la forte sensibilità al rumore e alle distorsioni sul colore introdotte nelle versione del codice da decodificare, proporzionalmente al numero di colori distinti utilizzati. In primis lo spazio colore RGB utilizzato per generare i QR Code è dipendente dal dispositivo: differenti dispositivi usati per catturare i colori, come le fotocamere, riconoscono i valori di R, G e B in modo diverso da produttore a produttore. In aggiunta, anche la qualità della fotocamera e le condizioni luminose condizionano in maniera importante la fase di cattura e rilevamento dei colori distinti, insieme alla fase di stampa che comporta interferenze tra canali colore (cfr. Sezione 4.2). Se il QR Code viene usato solamente nelle trasmissioni digitali, questi limiti legati alla fotocamera e alla stampante non vanno considerati.

4.4 Compressione dei dati

La compressione dei dati, che avviene durante la codifica di sorgente, permette di codificare le informazioni usando un minor numero di bit rispetto alla rappresentazione originale. Viene ridotta la ridondanza dei dati e quindi lo spazio occupato. In questo modo si riduce il consumo di risorse durante la fase di trasmissione: una dimensione inferiore genera un throughput più contenuto e un ritardo end-to-end più piccolo, oltre a un ridotto tempo di trasmissione.

Tale compressione può comportare o meno delle perdite di informazioni, così si distinguono le compressioni *lossy* o *lossless*. Le compressioni lossy eliminano i bit ritenuti trascurabili,

hanno un fattore di compressione alto e riducono significativamente lo spazio occupato dai dati, d'altro canto introducono una distorsione che non permette di recuperare esattamente il contenuto originale, ma solo con buona approssimazione. Invece le compressioni lossless rimuovono solo i bit che generano una ridondanza statistica, comprimendo in maniera meno aggressiva le informazioni e permettendo una ricostruzione perfetta del contenuto originale in fase di decompressione. Anche se il fattore di compressione risulta più piccolo e lo spazio occupato viene ridotto di poco. La scelta di una tipologia ha sicuramente dei compromessi da considerare, ma spesso si basa sul contenuto che deve essere compresso e sul relativo uso: per i testi solitamente si preferisce una codifica senza perdite perché ogni singolo carattere è importante nel contenuto, per le immagini invece si possono usare tranquillamente delle codifiche con perdite per diminuire la dimensione del salvataggio o del trasferimento.

Nei QR Code la compressione dei dati favorisce un incremento della capacità del codice, dato che la dimensione dei dati da codificare viene ridotta. In particolare si sfruttano delle compressioni lossless in modo tale da permettere una ricostruzione esatta, senza perdite, in fase di decodifica delle informazioni contenute. La fase di compressione può essere vista come un preprocessing dei dati, per rendere la relativa codifica più efficiente.

4.4.1 *Run-Length Encoding* (RLE)

Run-Length Encoding (RLE) è un algoritmo per comprimere i dati senza perdita di informazioni (lossless). RLE è uno degli algoritmi di codifica più semplici e facili da applicare, sviluppatosi nell'ambito di elaborazione delle immagini digitali.

Questo algoritmo permette di ridurre la dimensione occupata da una stringa in cui si ripetono sequenze di simboli identici, codificando tali ripetizioni con il simbolo (*run value*) e il numero di volte che esso viene ripetuto (*run count*). Esso ha delle buone prestazioni quando si elaborano dati con molte ripetizioni, ad esempio immagini con pochi colori e uniformi, o comunque con tutti i tipi di file in cui avvengono ripetizioni di stessi simboli.

In [17] si applica RLE per incrementare la dimensione dei dati immagazzinabili in un QR Code monocromatico, in modo tale da riuscire a contenere anche immagini e messaggi vocali. I file vengono compressi prima di essere codificati nei QR Code.

Codifica QR Code

Il processo di codifica avviene a partire dal file binario della risorsa da codificare. Esso consiste in una lunga sequenza di soli bit 1 e bit 0, su cui si applica l'algoritmo di codifica RLE. In

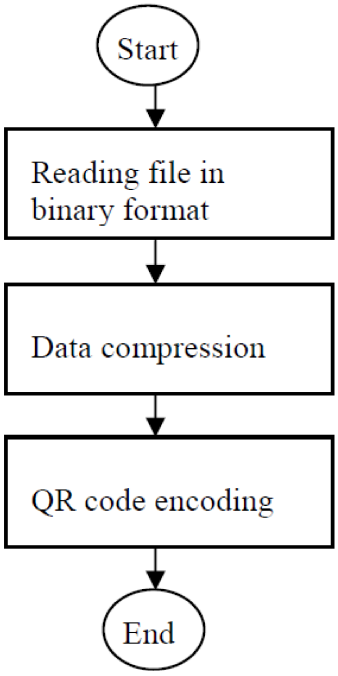


Figura 4.14: Diagramma di flusso della codifica di un QR Code con compressione [17]

```
a=double(bits);
rle(1)=a(1); m=2; rle(m)=1;
for i=1:length(a)-1
    if a(i)==a(i+1)
        rle(m)=rle(m)+1;
    else
        m=m+1; rle(m)=1;
    end
end
```

Figura 4.15: Algoritmo MATLAB per la codifica RLE [17]

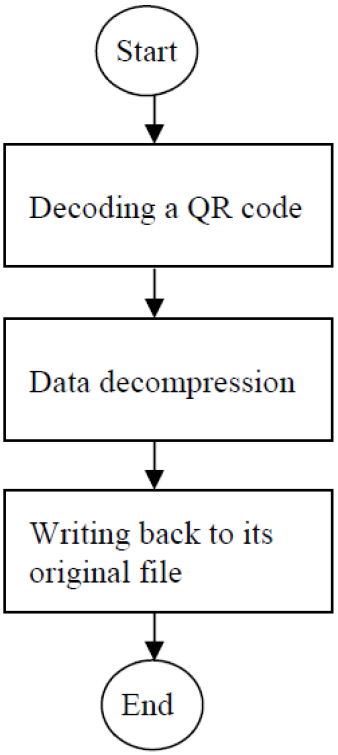


Figura 4.16: Diagramma di flusso della codifica di un QR Code con decompressione [17]

questo modo ogni stringa di 0 e 1 viene convertita nel rispettivo simbolo e contatore del numero di ripetizioni. In realtà, per effettuare una compressione ancora più efficace, dopo aver espresso il simbolo iniziale, si può omettere di indicare i successivi simboli, rappresentando solo la lunghezza delle ripetizioni: a partire dal bit iniziale, ci saranno sequenze di lunghezza specificata il cui valore del bit è alternativo rispetto alla sequenza precedente. Ad esempio,

$$111100000110010111 - (RLE) \rightarrow 14522113 \tag{4.5}$$

dove la prima ripetizione è del bit 1 e ha lunghezza 4, la successiva è del bit 0, sottinteso come valore alternativo rispetto alla precedente sequenza, con lunghezza 5, la successiva ancora è del bit 1, sottinteso, con lunghezza 2, e si procede in questo modo.

Infine, la stringa prodotta da RLE viene codificata secondo la modalità numerica prevista dallo Standard dei QR Code, e si completano i passi della codifica fino alla produzione del QR Code monocromatico.

Il processo descritto viene illustrato tramite un diagramma di flusso in Fig. 4.14 e in Fig. 4.15 viene riportato l'algoritmo in linguaggio MATLAB utilizzato per la codifica RLE.

```
rle=x;
l=0;
for t=2:length(rle)
    l=l+rle(t);
end
a=false(1,l); a(1)=rle(1);
m=2;n=1;
for i = 2:length(rle)-1
    for j=1:rle(m)
        a(n+1)=a(n); n=n+1;
    end
    m=m+1;a(n)=~a(n);
end
a=double(a);
```

Figura 4.17: Algoritmo MATLAB per la decodifica RLE [17]

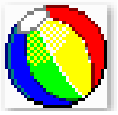


Figura 4.18: Immagine a colori da codificare con RLE [17]

Decodifica QR Code

Il processo inizia con l'acquisizione del QR Code attraverso una fotocamera e la conseguente decodifica per ricavare la sequenza di caratteri numerici, secondo i passi standard.

La stringa ricavata contiene le informazioni relative ai run count e ai run value, a cui viene applicato l'algoritmo di decodifica di RLE. Ricostruendo le ripetizioni dei bit, replicando alternativamente i simboli binari in accordo con la lunghezza specificata dal run value, si ottiene la sequenza binaria associata al file originale.

Il processo descritto viene illustrato tramite un diagramma di flusso in Fig. 4.16 e in Fig. 4.17 viene riportato l'algoritmo in linguaggio MATLAB utilizzato per la decodifica RLE.

Vantaggi

Grazie a questo approccio in cui è stata inclusa la fase di compressione dei dati, si sono estesi i limiti capacitivi del QR Code, permettendo di immagazzinare anche file vocali e grafici di notevoli dimensioni, senza limitarsi a caratteri testuali. Questa codifica non genera alcun tipo di perdita, permettendo una ricostruzione esatta del contenuto originale compresso.

Per verificare il metodo proposto, gli autori in [17], hanno fornito i risultati di un test in cui sono stati codificati due immagini nel QR Code: un'immagine a colori e una monocromatica.

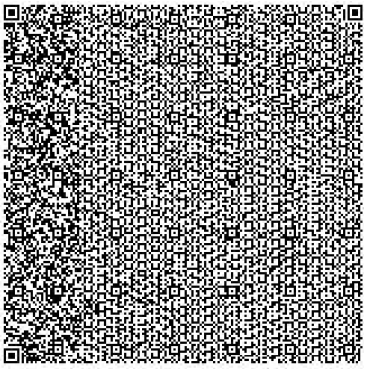


Figura 4.19: QR Code ottenuto dalla codifica con RLE di Fig. 4.18 [17]

Figura 4.20: Immagine monocromatica da codificare con RLE [17]

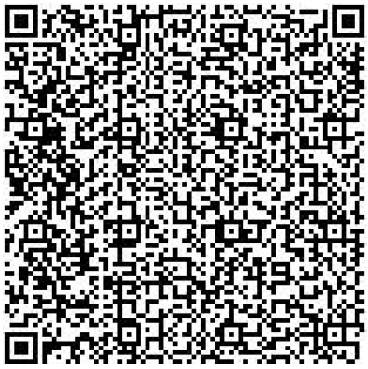


Figura 4.21: QR Code ottenuto dalla codifica con RLE di Fig. 4.20 [17]

La codifica con RLE ha compresso rispettivamente, nel primo caso (Fig. 4.18) e nel secondo caso (Fig. 4.20), 2912 caratteri in 1200 (rapporto di compressione: 2.43) e 6400 caratteri in 382 (rapporto di compressione: 16.75). L'immagine monocromatica è stata compressa maggiormente perché ha sequenze ripetute di bit più lunghe rispetto all'immagine a colori: infatti la prima è composta da solo 2 colori e risulta più semplice rispetto alla seconda con più colori diversi (almeno 6). In seguito alla compressione, sono stati codificati i 2 QR Code e si può notare che quello ottenuto dall'immagine a colori (Fig. 4.19) è maggiormente denso di moduli rispetto al corrispettivo monocromatico (Fig. 4.21), dato che la percentuale di compressione è minore.

La dimensione del file di entrambe le immagine è rimasta uguale prima della compressione e codifica nel QR Code, e dopo la decodifica e decompressione per riottenerle, senza alcuna perdita durante il processo.

Lo stesso test è stato effettuato su un file audio, che è stato correttamente compresso e codificato, e successivamente decodificato senza alcun tipo di perdita.

Infine, eccetto il processo di compressione e decompressione dei dati, tutti i passi della codifica e della decodifica coincidono con quanto già esistente per i QR Code standard (cfr. Capitolo 2).

Svantaggi

Il rapporto di compressione ottenuto dalla codifica RLE, e quindi l'aumento capacitivo del QR Code, risulta variabile in funzione dei dati forniti in ingresso: se ci sono lunghe sequenze di bit ripetuti il rapporto è molto elevato, come nel caso di un'immagine con pochi colori e molto omogenei, con poche sfumature. Nel caso in cui non siano presenti ripetizioni di bit sufficientemente lunghe, come nel caso di un'immagine con molti colori e sfumature, RLE fornisce un ridotto rapporto di compressione, provocando in alcuni casi addirittura un aumento della dimensione del file. Infatti, nell'esempio visto precedentemente RLE comprime l'immagine monocromatica con un rapporto di compressione di 16.75, quasi 7 volte superiore al rapporto di compressione di 2.43 ottenuto dall'immagine a colori.

Inoltre, per effettuare la compressione e la decompressione con RLE sono richiesti degli algoritmi aggiuntivi rispetto al tradizionale processo di codifica. Di conseguenza deve essere aggiunto uno strato di software ulteriore nel codificatore e nel decodificatore per permettere di comprimere i dati e recuperarli in seguito durante la scansione.

4.4.2 Altre codifiche lossless

Oltre a RLE, esistono numerose codifiche lossless per comprimere i dati.

In [13] viene proposta la combinazione della tecnica della compressione e del multiplexing dei QR Code. In particolare si fa riferimento alla codifica di Huffman e alla compressione in formato ZIP, applicate dopo aver mappato i caratteri da codificare nel corrispettivo valore ASCII trasformato in binario. I dati compressi vengono poi codificati nel QR Code ottenuto attraverso una fase di multiplexing.

La codifica di Huffman produce un codice a prefisso con parole a lunghezza variabile, che risulta essere ottimo, cioè con lunghezza media della parole $\bar{L} = H + 1$, dove H è l'entropia della sorgente binaria (in questo caso il file di caratteri). I simboli (caratteri) più frequenti vengono associati a parole di codice più corte, mentre quelli meno frequenti a parole di codice più lunghe. La procedura avviene in due fasi: nella prima parte si crea l'albero di Huffman, mentre nella seconda parte viene attraversato per localizzare i simboli.

La compressione ZIP solitamente prevede l'utilizzo del formato compresso Deflate, inventato da Phil Katz e specificato dalla RFC 1951 (1996). I dati contenuti consistono in una serie di blocchi compressi usando una combinazione dell'algoritmo LZ77 e della codifica di Huffman. Ogni blocco contiene una parte con una coppia di alberi di Huffman per descrivere la rappresentazione dei dati compressi e un'altra parte con i dati compressi con LZ77. Quest'ultima è composta da due tipi di elementi: literal di stringhe che non hanno duplicati nei precedenti 32 KB di dati, e puntatori, che sostituiscono stringhe duplicate nei precedenti 32 KB, rappresentati da una coppia (lunghezza, distanza), dove la lunghezza limite è di 258 B e la distanza limite è di 32 KB. I due alberi di Huffman, compressi essi stessi usando la codifica di Huffman, rappresentano separatamente in due codici i literal insieme alle lunghezze, e le distanze.

Dai test effettuati in [13], data in ingresso una quantità di dati pari a 3 KB, la compressione con Huffman restituisce una dimensione di 1.2 KB, invece la compressione ZIP restituisce una dimensione di 1 KB, risultando quindi la più efficiente.

Capitolo 5

Analisi di dataset di QR Code

La struttura dei QR Code è progettata per permettere una scansione dei codici da tutte le angolazioni possibili, senza richiedere alcun tipo di allineamento. Questo significa che un decodificatore è in grado di identificare correttamente ed estrarre i dati da QR Code ruotati di un certo angolo, distorti e danneggiati. Inoltre, dato l'uso di moduli solo di colore bianco e nero, i QR Code monocromatici sono più rubusti alle diverse condizioni luminose, rispetto ad esempio a una codifica con numerosi colori.

In seguito si vogliono analizzare diversi dataset di QR Code disponibili online, che raccolgono numerose immagini di QR Code sottoposte a vari trattamenti di distorsione e rumore. In questo modo si vogliono valutare le prestazioni del processo di decodifica e i parametri che lo influenzano maggiormente.

Nell'analisi vengono sfruttate due librerie Python:

- *pyzbar*[1]: lettore di codici a barre monodimensionali e QR Code, basato sulla libreria open source *ZBar*, usato per la decodifica dei QR Code;

- *opencv-python*[2]: wrapper della libreria open source *OpenCV*, ampiamente diffusa nell'ambito della computer vision, usato per l'elaborazione delle immagini in input al decodificatore.

La decisione riguardo la libreria per la decodifica dei QR Code è stata presa valutando diverse librerie in Python, tra cui *OpenCV* e *ZXing*, scegliendo quella con prestazioni migliori.

[1] https://pypi.org/project/pyzbar/
[2] https://pypi.org/project/opencv-python/

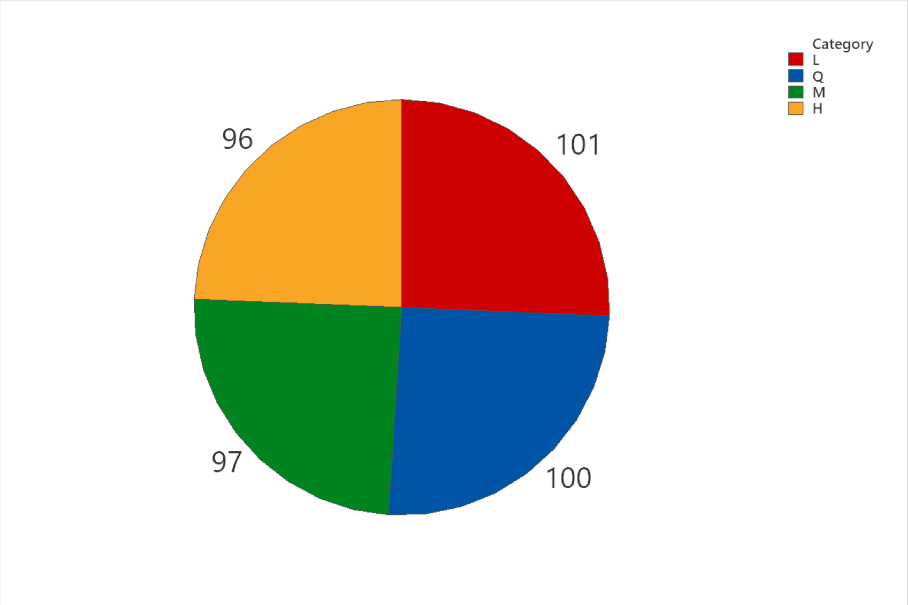


Figura 5.1: Frequenza dei livelli di correzione degli errori nel dataset 1

5.1 Dataset 1

Il dataset considerato è disponibile in [5].

5.1.1 Descrizione

Il dataset contiene 394[3] immagini di QR Code standard (moduli bianchi e neri) in formato *jpg* (compressione JPEG), di dimensione 2560 x 1440 pixel, scattate dalla fotocamera di uno smartphone Nokia N900 (5 Megapixel). Il dataset è descritto da un file *csv*, che riporta per ogni immagine i dati codificati, il livello di correzione degli errori del QR Code, l'area occupata dal codice e le distorsioni presenti.

I QR Code codificano caratteri alfabetici maiuscoli e minuscoli che compongono frasi prese da un insieme di 50 citazioni d'autore, la cui lunghezza è compresa tra 46 e 221 simboli. Perciò viene adottata la modalità di codifica byte e il numero di moduli richiesti, variabili in base alle dimensione delle informazioni codificate, comporta la generazione di QR Code compresi tra la versione 3 e la versione 16. Il livello di correzione degli errori adottato coinvolge tutte le 4 categorie esistenti L, M, Q e H equamente, come mostrato in Fig. 5.1.

Le immagini nel dataset sono scatti acquisiti dalla realtà, non sono state sintetizzate da alcun codificatore. In particolare nelle foto vengono catturati dei fogli contenenti al centro un singolo QR Code, avvolto su tutti i lati da numerose scritte. Il QR Code occupa un'area

[3]In realtà le immagini presenti nel dataset 1 sono 410, ma le informazioni di 16 foto risultano assenti, quindi sono state escluse dall'analisi seguente.

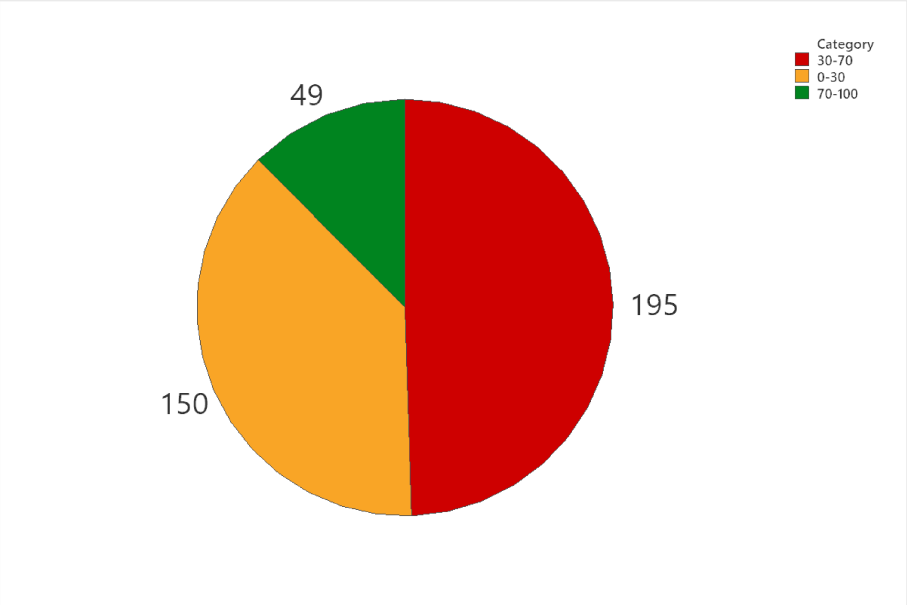


Figura 5.2: Frequenza dell'area occupata dai QR Code nel dataset 1

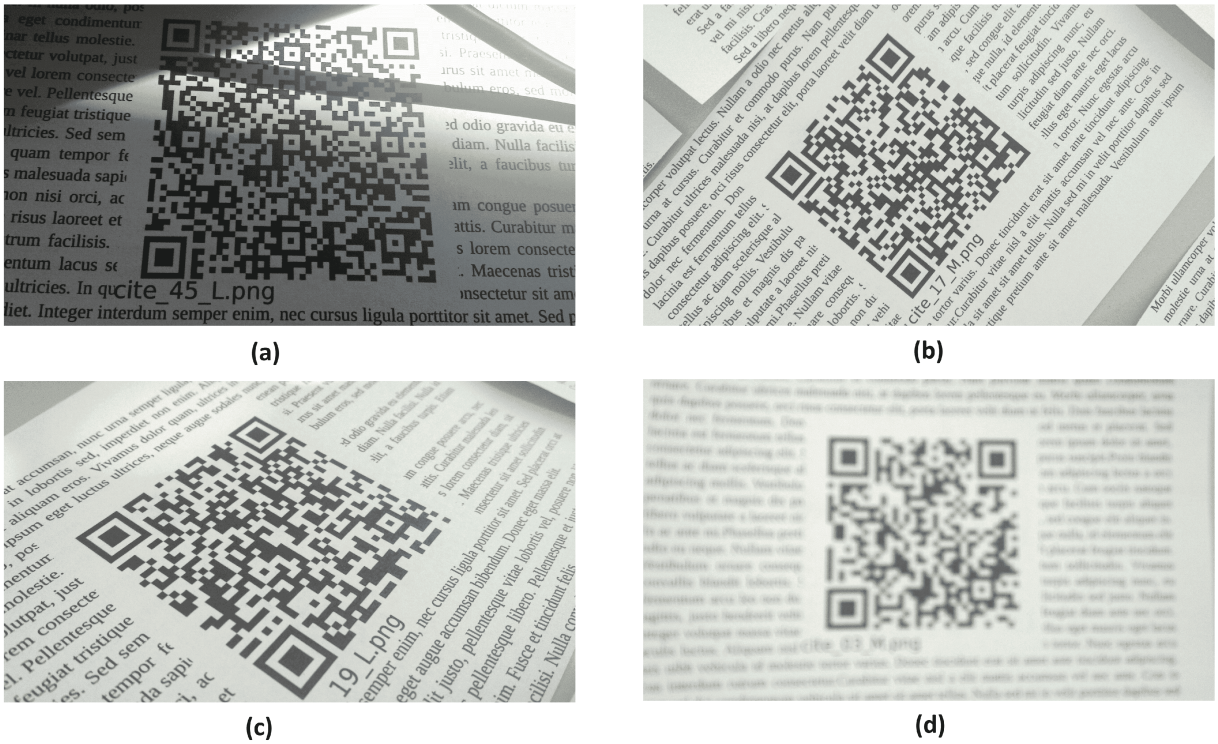


Figura 5.3: Esempi di QR Code nel dataset 1 con (a) illuminazione irregolare, (b) rotazione, (c) distorsione prospettica e (d) sfocatura

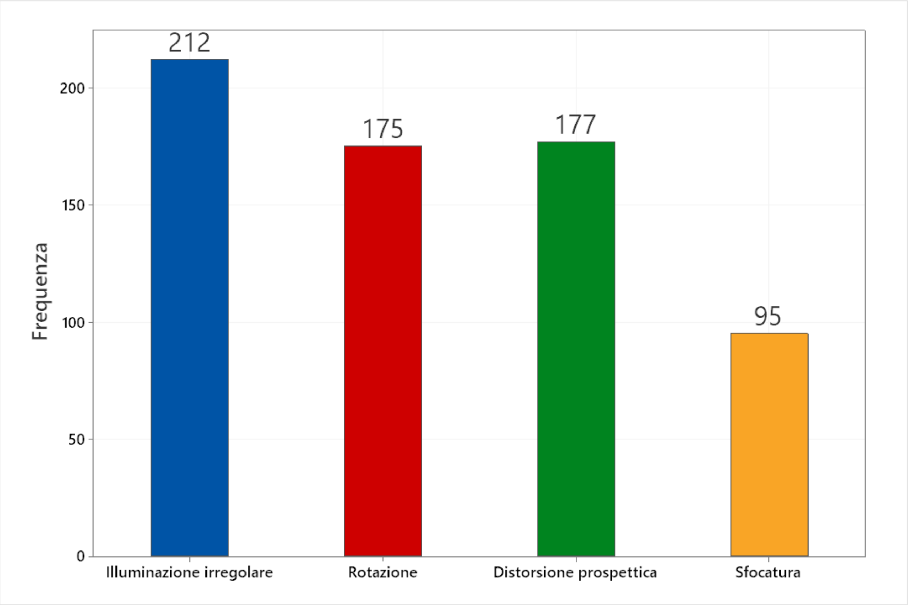


Figura 5.4: Frequenza delle tipologie di distorsione presenti nel dataset 1

di dimensione variabile all'interno dell'immagine, in base alla distanza a cui è avvenuto lo scatto: vengono identificati 3 intervalli di copertura della superficie dell'immagine da parte del codice. Prevalentemente i QR Code occupano un'area abbastanza ridotta, pari o inferiore al 70% della foto, come si può notare in Fig. 5.2 dove viene illustrata la distribuzione di tali intervalli all'interno del dataset.

Le condizioni in cui avvengono gli scatti rappresentano una buona panoramica delle distorsioni che normalmente interessano i QR Code: essi sono acquisiti ruotati, in prospettiva, distorti, sfuocati e colpiti da una luce intensa così da creare un'illuminazione non costante con zone chiare e scure come mostrato in Fig. 5.3. Tutte le distorsioni, ad esclusione della rotazione, presentano un livello di intensità diverso nei diversi casi. Si possono raggruppare le tipologie di distorsione applicate alle immagini, riportate nel file csv allegato al dataset, nelle seguenti categorie:

- luminosità irregolare
- rotazione
- distorsione prospettica
- sfocatura

Nella Fig. 5.4 è illustrato un grafico che rappresenta la frequenza di ciascuna tipologia di distorsione presente nei QR Code contenuti nelle immagini del dataset 1, si tenga in considerazione che possono essere applicate in contemporanea più distorsioni.

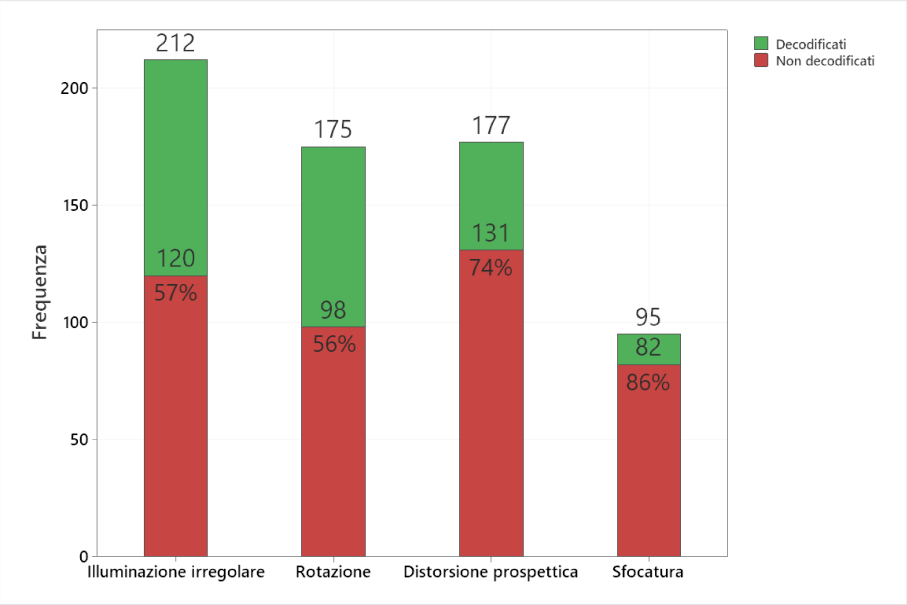


Figura 5.5: Frequenza di ogni tipologia di distorsione nei QR Code non decodificati, rapportata percentualmente sulla frequenza totale dei casi della relativa tipologia di distorsione, nel dataset 1

5.1.2 Risultati

All'inizio le foto dei QR Code vengono caricate dal rispettivo file jpg attraverso la libreria opencv, che decodifica l'immagine catturata da una fotocamera con sensore RGB in una matrice pluridimensionale di valori corrispondenti ai canali colore, nell'ordine, B, G e R, poi si effettua una conversione nello spazio colore a livelli di grigi. Il decodificatore implementato da pyzbar riceve i dati relativi alla matrici e compie il processo di decodifica, estraendo i dati codificati.

Dai risultati ottenuti dall'analisi delle immagini a dimensioni originali, confrontati con i dati originariamente codificati (riportati nel file csv), emerge che sono state decodificate correttamente 126 immagini su 394 totali, che corrisponde a un tasso di decodifica corretta del 32%. Tuttavia, poiché tra le rimanenti 268 immagini, che non è stato possibile decodificare, sono presenti circa una decina di casi in cui le distorsioni applicate consistono solamente in una rotazione, allora si riesegue il test provando a ridurre le dimensioni al 55% delle dimensioni originali delle immagini, passando da 2560 x 1440 a 1408 x 792 pixel.

Con questa scelta aumentano le prestazioni del test, infatti avviene una corretta decodifica di 195 immagini di QR Code su 394, cioè circa il 50% del dataset. Evidentemente le foto originali sono esposte a del rumore eccessivo per via dell'alta risoluzione o ad artefatti causati dalla compressione JPEG, che hanno impedito di completare opportunamente la decodifica dei dati contenuti nel QR Code: il ridimensionamento ha aiutato a ridurre il rumore dato che più pixel vengono fusi in uno solo, mitigando così i valori più rumorosi.

Analizzando le statistiche ricavate dalle immagini che non sono state decodificate correttamente, si può notare che nella maggior parte dei casi hanno pesato principalmente la distorsione prospettica e la sfocatura. Infatti, come emerge dal grafico in Fig. 5.5, i casi di QR Code non decodificati presentano una maggiore frequenza, confrontata sul totale dei casi, di distorsioni dovute alla prospettiva e alla sfocatura: in particolare quest'ultima vede quasi la totalità dei casi (86%) in cui è stata applicata non essere interpretata. In secondo luogo hanno contribuito anche l'illuminazione irregolare e la rotazione dei QR Code. Tuttavia si tratta di un'analisi qualitativa, che non considera i diversi livelli di intensità della distorsione applicata o la presenza contemporanea di tipologie multiple.

Analizzando allora manualmente le immagini non decodificate, una alla volta, trova conferma il fatto che la sfocatura presente nei casi interessati, a volte provocata da un effetto mosso, è in media di intensità elevata. Di conseguenza l'assenza di messa a fuoco e di nitidezza dell'immagine crea un forte rumore nella scansione, alterando i colori e i confini dei moduli, impedendo quindi una demodulazione corretta da parte del decodificatore.

Per quanto riguarda la distorsione prospettica, nei casi in cui il livello di intensità è elevato, essa sicuramente crea dei problemi di dimensionamento del QR Code per via delle deformazioni introdotte, causando quindi un'errata localizzazione dei pattern funzionali e delle posizioni dei moduli in termini di coordinate, impedendo così il proseguimento della fase di decodifica. Essa è presente in diversi casi, ma di rado singolarmente: molto spesso è accompagnata da condizioni luminose non costanti e da un'area occupata dal codice abbastanza ridotta, che contribuiscono a generare rumore. Allo stesso modo anche la sola rotazione del QR Code non ha creato disturbi nella decodifica, quando però viene combinata con altre distorsioni nascono i problemi.

Inoltre, diversamente da quando evidenziato delle statistiche, dalle immagini non decodificate emerge che il rumore dovuto alle condizioni di luce irregolari è risultato veramente invasivo: la presenza di un estremo contrasto tra le zone in ombra e le zone illuminate sulla superficie del QR Code ha impedito all'algoritmo di thresholding del decodificatore, durante la fase di demodulazione, di trovare un valore di soglia adatto per compiere l'elaborazione dell'immagine.

Nell'analisi delle foto non decodificate si ha considerato anche l'impatto di fattori quali la copertura della superficie dell'immagine da parte del QR Code e il relativo livello di correzione degli errori, ma non si è manifestata alcuna tendenza interessante. Molto probabilmente le distorsioni hanno avuto un peso più significativo: a questo punto allora conviene utilizzare un livello di correzione degli errori più basso per aumentare la quantità di caratteri codificabili (a parità di versione e modalità di codifica), soprattutto nei casi in cui l'intensità della distorsione

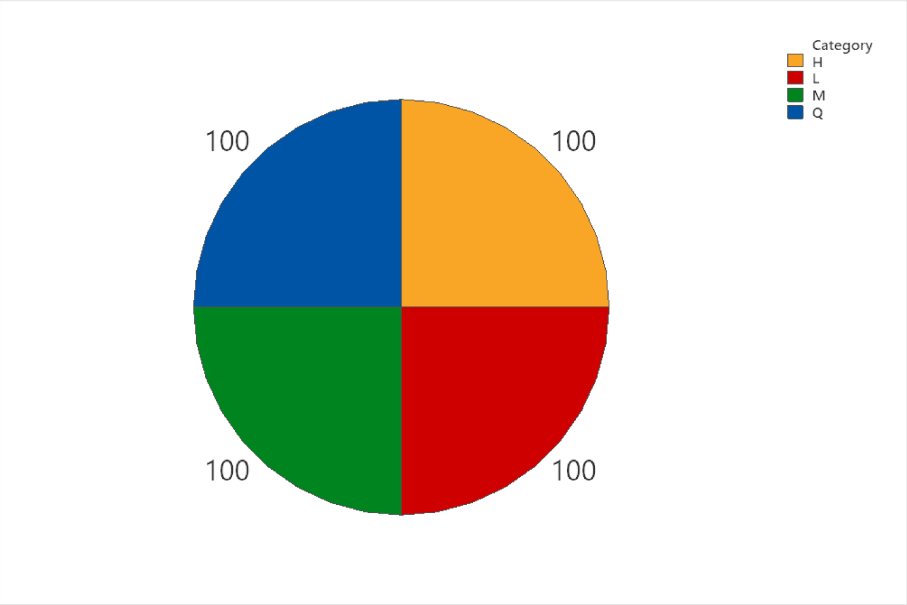


Figura 5.6: Frequenza dei livelli di correzione degli errori nel dataset 2

è contenuta.

In conclusione si può affermare che il processo di decodifica, a partire dal dataset considerato, ha mostrato dei risultati poco incoraggianti, fermandosi a un tasso di corretta decodifica del 50% dopo il ridimensionamento. I principali protagonisti dell'errata interpretazione dei QR Code sono la luminosità irregolare e la sfocatura, quando introducono importanti distorsioni. Serve quindi effettuare del preprocessing sulle immagini per correggere ulteriormente le distorsioni, rispetto a quanto fatto dal decodificatore: ad esempio l'utilizzo di algoritmi di thresholding adattivi più complessi, che si adattano localmente alle condizioni luminose, oppure applicare la convoluzione con filtri appositi per rimuovere la sfocatura.

5.2 Dataset 2

Il dataset considerato è disponibile in [5].

5.2.1 Descrizione

Il dataset contiene 400 immagini di QR Code standard (moduli bianchi e neri) in formato *jpg* (compressione JPEG), di dimensione 604 × 402 pixel, scattate da una fotocamera. Il dataset è descritto da un file *csv*, che riporta, per ogni immagine, i dati codificat, il livello di correzione degli errori del QR Code, l'area occupata dal codice e le distorsioni presenti.

I QR Code codificano caratteri alfabetici maiuscoli e minuscoli che compongono frasi prese da un insieme di 50 citazioni d'autore, la cui lunghezza è compresa tra 46 e 221 simboli, come

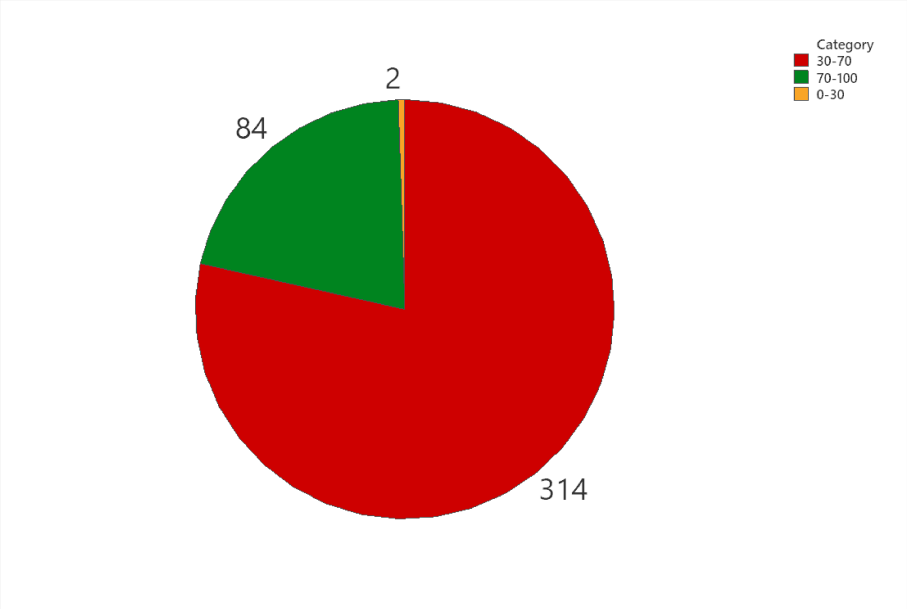


Figura 5.7: Frequenza dell'area occupata dai QR Code nel dataset 2

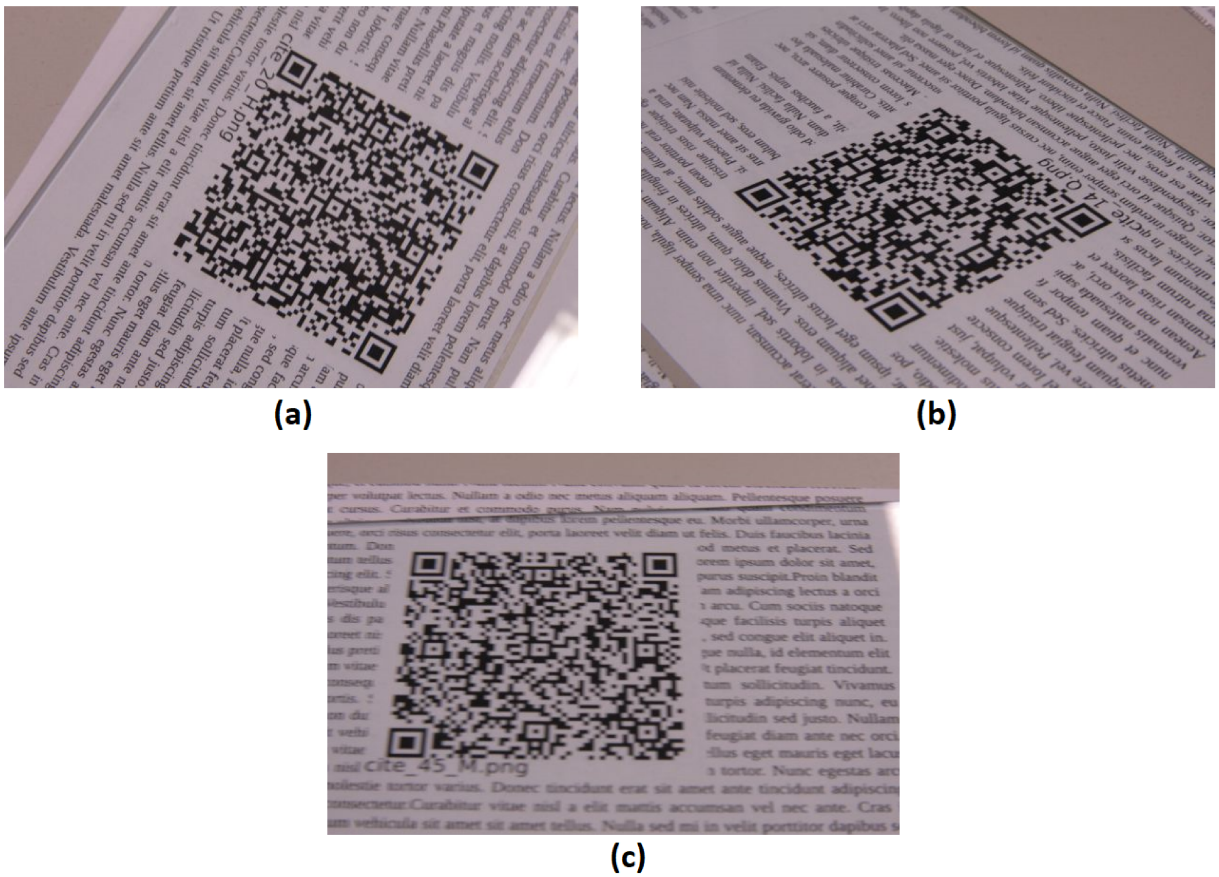


Figura 5.8: Esempi di QR Code nel dataset 2 con (a) rotazione, (b) distorsione prospettica e (c) sfocatura

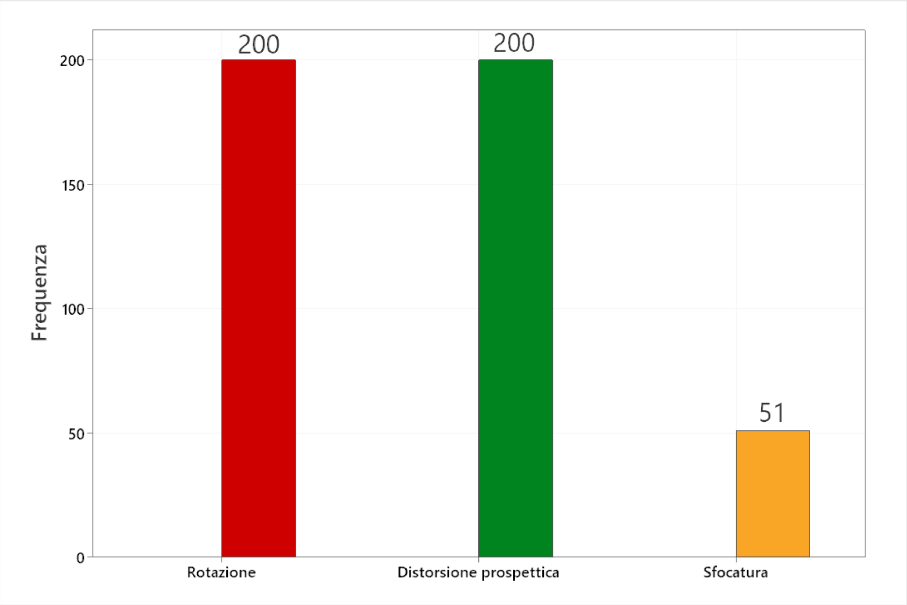


Figura 5.9: Frequenza delle tipologie di distorsione presenti nel dataset 2

in precedenza. Perciò viene adottata la modalità di codifica byte e i QR Codes sono compresi tra la versione 3 e la versione 16. Il livello di correzione degli errori adottato coinvolge tutte le 4 categorie esistenti L, M, Q e H equamente, come mostrato in Fig. 5.6.

Allo stesso modo del dataset 1, le immagini in questo dataset sono scatti acquisiti dalla realtà. In particolare nelle foto vengono catturati dei fogli contenenti al centro un singolo QR Code, avvolto su tutti i lati da numerose scritte. Il QR Code occupa un'area di dimensione variabile all'interno dell'immagine, in base alla distanza a cui è avvenuto lo scatto: vengono identificati 3 intervalli di copertura della superficie dell'immagine da parte del codice. Prevalentemente i QR Code occupano un'area abbastanza ridotta, tra il 30% e il 70% della foto, come si può notare in Fig. 5.7 dove viene illustrata la distribuzione di tali intervalli all'interno del dataset.

Le condizioni in cui avvengono gli scatti introducono distorsioni che normalmente interessano i QR Code: essi sono acquisiti ruotati, in prospettiva, distorti e sfuocati come mostrato in Fig. 5.8. A differenza del dataset analizzato in Sezione 5.1, non viene introdotta la distorsione causata dall'illuminazione irregolare, anche se, in generale, le immagini sono state acquisite tutte in condizione di bassa luminosità. Tutte le distorsioni, ad esclusione della rotazione, presentano un livello di intensità diverso nei diversi casi. Si possono raggruppare le tipologie di distorsione applicate alle immagine, riportate nel file csv allegato al dataset, nelle seguenti categorie:

- rotazione
- distorsione prospettica

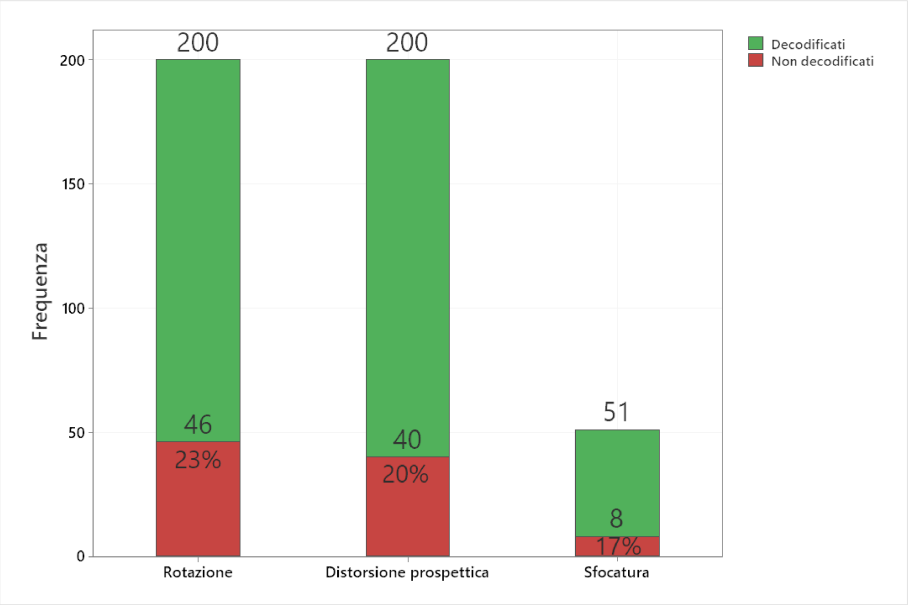


Figura 5.10: Frequenza di ogni tipologia di distorsione nei QR Code non decodificati, rapportata percentualmente sulla frequenza totale dei casi della relativa tipologia di distorsione, nel dataset 2

- sfocatura

Nella Fig. 5.9 è illustrato un grafico che rappresenta la frequenza di ciascuna tipologia di distorsione presente nei QR Code contenuti nelle immagini del dataset 2, si tenga in considerazione che possono essere applicate in contemporanea più distorsioni.

5.2.2 Risultati

Dopo aver trasformato le immagini in matrici di valori attraverso opencv, allo stesso modo descritto in Sezione 5.1, vengono elaborate dal decodificatore di pyzbar. I risultati ottenuti, confrontati con i dati originariamente codificati (riportati nel file csv), evidenziano che 349 su 400 immagini di QR Code sono state correttamente decodificate, raggiungendo così un tasso di decodifica corretta pari al 87%.

Analizzando le statistiche mostrate in Fig. 5.10 riguardanti i casi di QR Code non decodificati, si nota l'importante impatto della distorsione dovuta alla rotazione e alla deformazione prospettica. In realtà, analizzando attentamente i singoli casi, questi due tipi di distorsione appaiono prevalentemente insieme, quasi mai singolarmente. Solo in una decina di casi sono presenti le distorsioni di sfocatura.

Procedendo con l'analisi delle singole immagini non decodificate, nella maggior parte appare che molte delle distorsioni prospettiche causano un forte allungamento in direzione diagonale, che deforma la dimensione e la forma del codice: questo elemento, insieme alla rotazione,

comporta un'errata localizzazione degli elementi del QR Code. In aggiunta, in tutto il dataset è presente un livello di luce molto basso, che contribuisce ad aumentare l'effetto di disturbo provocato dal rumore presente nell'immagine e di conseguenza corrompe ulteriormente la scansione del QR Code fatta dal decodificatore, già distorta a causa della prospettiva e della rotazione.

Tra tutti i casi di sfocatura nel dataset, quasi una decina non è stata decodificata per via dell'intensa distorsione applicata: i bordi dei moduli del codice e di tutti gli elementi costitutivi, specialmente i finder pattern che ne permettono la localizzazione, non sono più distinguibili, sono rilevabili solo chiazze di colore.

Nell'analisi delle foto non decodificate si ha considerato anche l'impatto di fattori quali la copertura della superficie dell'immagine da parte del QR Code e il relativo livello di correzione degli errori, ma non si è manifestata alcuna tendenza interessante. Molto probabilmente le distorsioni hanno avuto un peso più significativo: a questo punto allora conviene utilizzare un livello di correzione degli errori più basso per aumentare la quantità di caratteri codificabili (a parità di versione e modalità di codifica), soprattutto nei casi in cui l'intensità della distorsione è contenuta.

In conclusione si può affermare che il processo di decodifica, a partire dal dataset considerato, ha mostrato degli ottimi risultati, sfiorando un tasso di corretta decodifica del dataset pari a 90%. Infatti il livello di intensità delle distorsioni applicate è stato in generale contenuto, oltre al fatto che non era presente un'illuminazione irregolare. Invece il principale fattore che ha favorito un'errata interpretazione dei QR Code è la combinazione di deformazione prospettica e rotazione, nei casi in cui ha introdotto importanti distorsioni; unitamente è intervenuto anche il rumore favorito dalle condizioni di bassa illuminazione nel dataset. Serve quindi effettuare del preprocessing sulle immagini per correggere ulteriormente le distorsioni, rispetto a quanto fatto dal decodificatore: ad esempio l'utilizzo di software per raddrizzare la prospettiva dei QR Code nelle foto oppure l'applicazione di tecniche per rimuovere il rumore nelle foto.

5.2.3 Confronto tra dataset 1 e 2

I due dataset analizzati contengono immagini molto simili: essi presentano la stessa tipologia di QR Code, lo stesso ambiente in cui si trovano i QR Code all'interno dell'immagine e lo stesso pool da cui sono presi i dati da codificare. Si differenziano per la risoluzione delle immagini, il dispositivo di acquisizione, e in particolare per le condizioni in cui viene lo scatto e le tipologie di distorsione presenti.

Per quanto riguarda le caratteristiche delle immagini contenute, sicuramente il dataset 1 contiene foto con risoluzione maggiore e illuminazione più elevata (tralasciando i casi in cui le condizioni luminose sono irregolari), mentre il dataset 2 presenta foto di dimensioni più ridotte e allo stesso tempo con un livello di illuminazione molto basso. Le distorsioni applicate e la relativa intensità variano significativamente tra i due dataset e tra i casi stessi.

Il primo fattore che emerge dai risultati ottenuti nei due dataset è la differenza nel tasso di decodifica corretta: nel primo si è fermato solo al 50%, mentre nel secondo ha raggiunto un valore pari a 87%. Il motivo principale è l'assenza nel dataset 2 di distorsioni causate da un'illuminazione irregolare, che si traduce di conseguenza su un'applicazione meno disturbata dell'algoritmo di thresholding. Inoltre, nonostante nei due dataset sia presente all'incirca la stessa quantità di immagini distorte a causa della sfocatura, nel dataset 1 c'è un numero più elevato di immagini non decodificate perché l'intensità della sfocatura è maggiore.

Continuando nell'analisi dei casi non correttamente decodificati, in entrambi i dataset hanno influito molto le distorsioni legate alla deformazione prospettica, dato che provocano l'alterazione della geometria del QR Code e rendono più difficile la localizzazione del codice. Un contributo proviene anche dalla rotazione, che impatta la corretta decodifica del QR Code soprattutto quando è usata in concomitanza ad altre tipologie di distorsioni, spesso quella prospettica e luminosa.

In conclusione, i due dataset visti hanno ben rappresentato le casistiche di distorsione più frequenti nella realtà quotidiana, ovvero, trasposto nell'ambito delle telecomunicazioni, il rumore che affligge la trasmissione nel canale di comunicazione. La struttura dei QR Code ha fornito un adeguato livello di robustezza, grazie in particolare ai pattern funzionali, tra cui i finder pattern, in fase di localizzazione, ai moduli bianchi e neri in fase di demodulazione e ai codici RS in fase di decodifica per la correzione degli errori. Sicuramente emerge una risposta del decodificatore diversa in base al rumore presente: maggiore è l'intensità delle distorsioni, anche considerando la specifica tipologia di disturbo presente, più si complica la fase di demodulazione al lato "ricevitore", cioè la corretta decisione dei dati codificati alla sorgente basandosi su quelli ricevuti con il rumore. Sicuramente condizioni irregolari di illuminazione e la distorsione prospettica, meno verosimilmente nella realtà la sfocatura, influenzano molto l'interpretazione dei codici, soprattutto quando l'intensità del rumore introdotto è elevata, altrimenti nei casi meno disturbati sono state corrette in maniera ottima. Si deve tener conto che la fase di elaborazione dell'immagine, fondamentale per il passo di demodulazione, varia in base all'implementazione del decodificatore utilizzato, adottando tecniche diverse di preprocessing, di thresholding e rilevazione del colore, con lo scopo ulteriore di correggere le interferenze

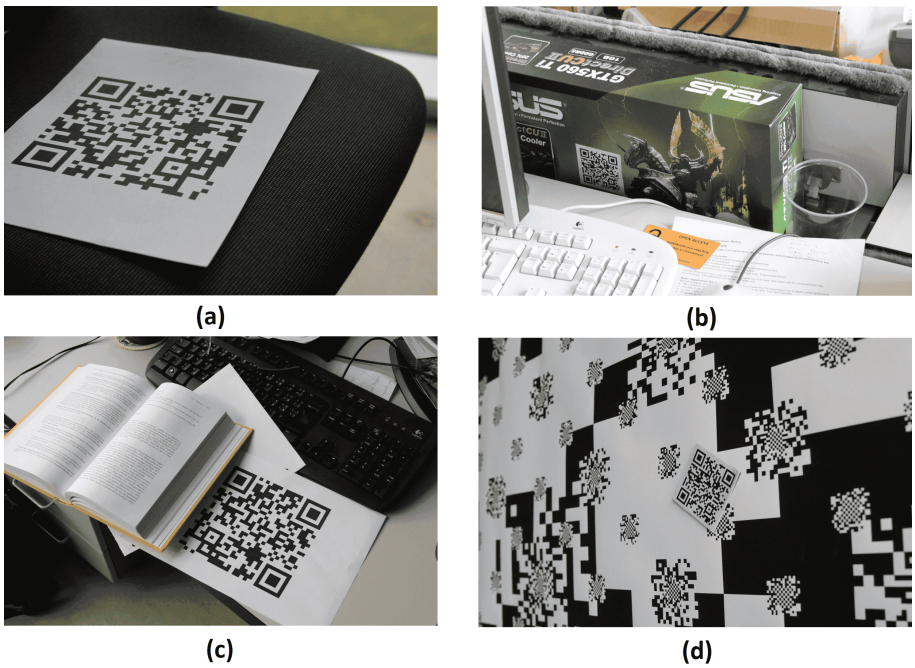


Figura 5.11: Esempi del dataset 3 dalla parte con QR Code singoli con (a) distorsione prospettica in *one_low*, (b) distorsione prospettica in *one_medium*, (c) illuminazione irregolare in *one_medium* e (d) sfondo rumoroso in *one_hard*

luminose.

5.3 Dataset 3

Il dataset considerato è disponibile in [11][4].

5.3.1 Descrizione

Il dataset può essere suddiviso in due parti: la prima parte contiene immagini in cui è presente un solo QR Code, mentre la seconda parte contiene immagini che singolarmente catturano numerosi QR Code. Ogni foto è descritta da un file *csv*, che riporta la coordinate della posizione del QR Code, le distorsioni presenti e i dati codificati.

Porzione di immagini con QR Code singoli

La porzione di dataset considerata contiene in totale 29 immagini di QR Code standard (moduli bianchi e neri) in formato *jpg* (compressione JPEG), di dimensione 4752 × 3168 pixel, scattate da una fotocamera Canon EOS 50D (15.1 Megapixel).

[4]Non sono stati considerati i gruppi *img_1080p* e *lightning* poichè contengono immagini ripetute dai dataset precedenti o dal dataset stesso.

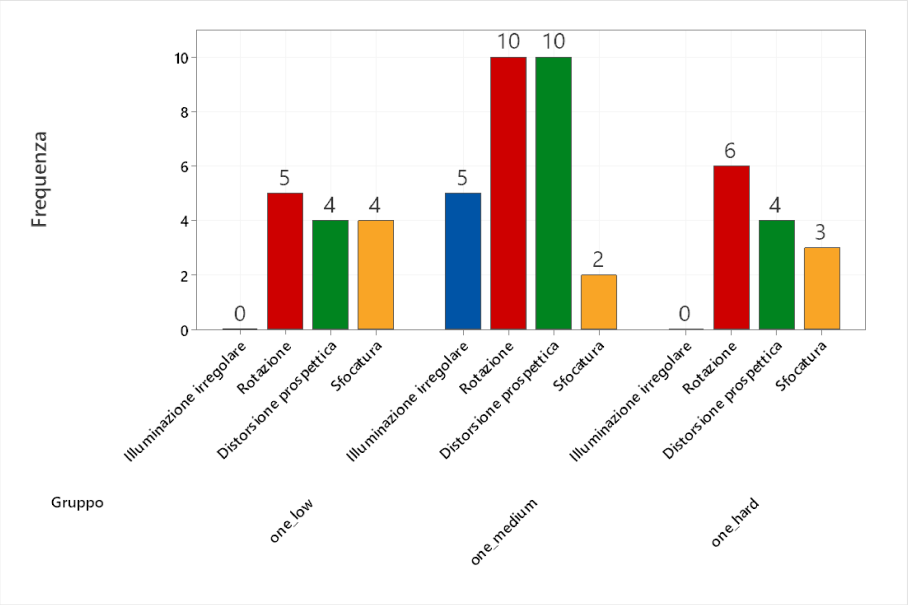


Figura 5.12: Frequenza delle tipologie di distorsione presenti nel dataset 3 nella parte con QR Code singoli

I QR Code codificano caratteri alfabetici maiuscoli e minuscoli che compongono frasi scelte tra 2 citazioni d'autore, la cui lunghezza è di 48 e 49 simboli. Perciò viene usata la modalità di codifica byte e il numero di moduli richiesti, in entrambi i casi di dati codificati, comporta la generazione di QR Code di versione 3-L.

Le immagini nel dataset vengono distribuite in 3 gruppi *one_low* (8), *one_medium* (13) e *one_hard* (8): essi si differenziano nella tipologia di distorsioni introdotte e, a differenza dei dataset precedenti, nell'ambiente in cui sono disposti i QR Code nell'immagine.

Il gruppo *one_low* contiene QR Code facilmente individuabili e decodificabili, che presentano delle rotazioni e delle minime distorsioni prospettiche e sfocature, in alcuni casi l'area occupata nell'imagine dal codice è molto ridotta. Il gruppo *one_medium* contiene QR Code maggiormente distorti rispetto ai precedenti, e nei quali viene introdotta in alcuni casi un'illuminazione irregolare; oltretutto essi sono inseriti in un'immagine con un ambiente meno sgombro, con più ostacoli per rendere più complicata la localizzazione del QR Code. Nel gruppo *one_hard* si raggiunge l'apice della difficoltà nell'identificazione dei QR Code, dove nello sfondo delle immagini sono presenti dei simboli che assomigliano ai QR Code, soprattutto nella forma e nella disposizione dei moduli, con lo scopo di ingannare il decodificatore. Alcuni esempi di immagini sono riportati in Fig. 5.11.

In Fig. 5.12 è illustrato un grafico che rappresenta, per ogni gruppo, la frequenza di ciascuna tipologia di distorsione presente nei QR Code contenuti nelle immagini della porzione con QR Code singoli del dataset 3, si tenga in considerazione che possono essere applicate in

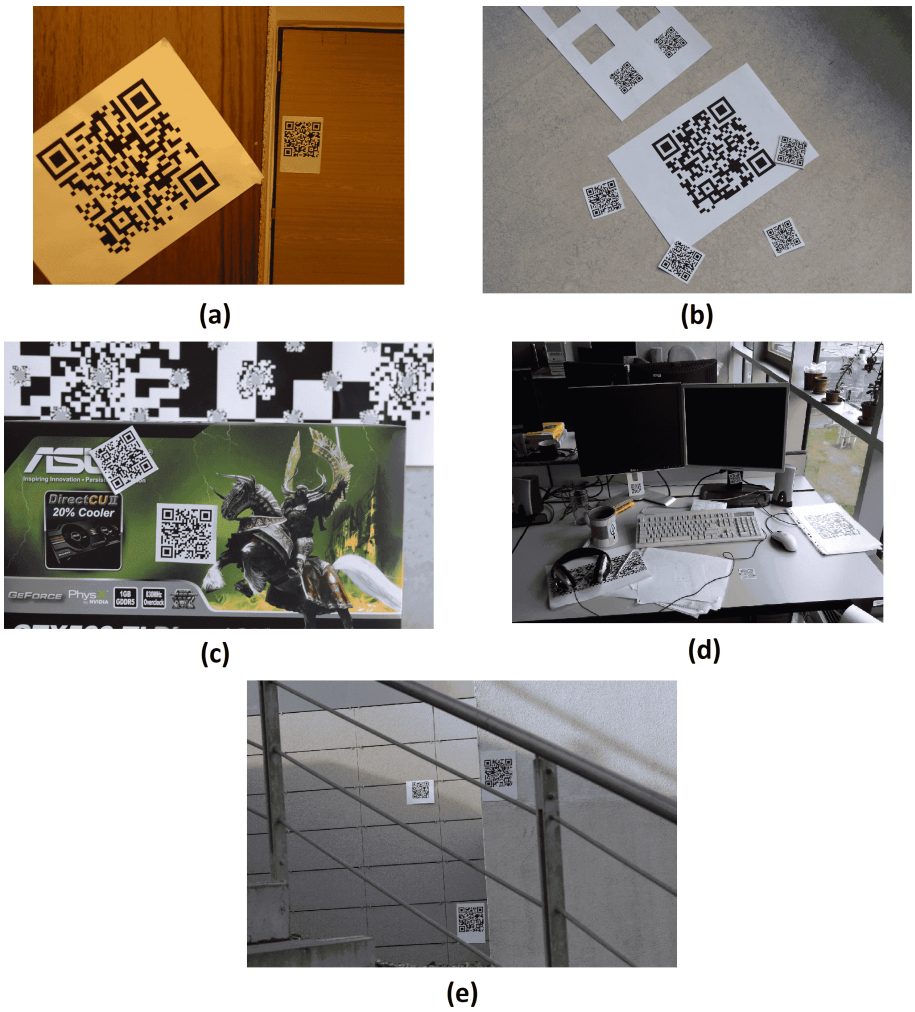


Figura 5.13: Esempi di QR Code multipli nel dataset 3 con (a) rotazione e sfocatura in *mobile_multi*, (b) rotazione in *more_low*, (c) sfondo rumoroso in *more_medium*, (d) sfondo rumoroso in *more_high* e (e) illuminazione irregolare in *semi_high*

contemporanea più distorsioni.

Porzione di immagini con numerosi QR Code

La porzione di dataset considerata contiene 52 immagini di QR Code standard (moduli bianchi e neri) in formato *jpg* (compressione JPEG), in parte di dimensione 4752 × 3168 pixel o inferiore, scattate da una fotocamera Canon EOS 50D (15.1 Megapixel) e l'altra parte di dimensione 2560 x 1920 o inferiore scattate dalla fotocamera di uno smartphone HTC Desire (5 Megapixel). In totale sono presenti 182 QR Code, dato che ogni immagine contiene più di un singolo QR Code, come riportato nel grafico in Fig. 5.14 rappresentante la frequenza del numero di codici contenuti nelle immagini.

I QR Code codificano caratteri alfabetici maiuscoli e minuscoli che compongono frasi scelte tra 2 citazioni d'autore, la cui lunghezza è di 48 e 49 simboli, oppure l'URL di un sito web lungo

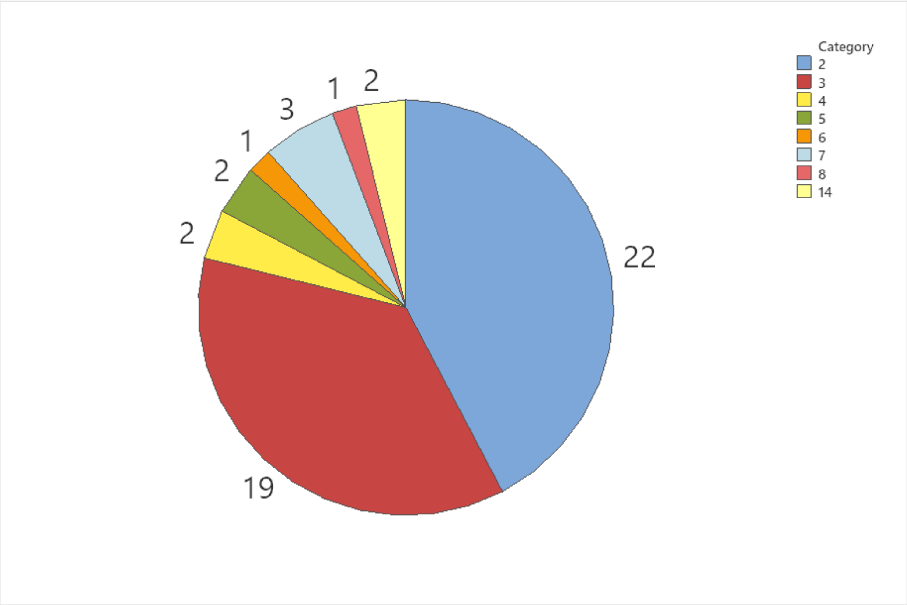


Figura 5.14: Frequenza del numero di codici contenuti singolarmente nelle immagini nel dataset 3 nella parte con QR Code multipli

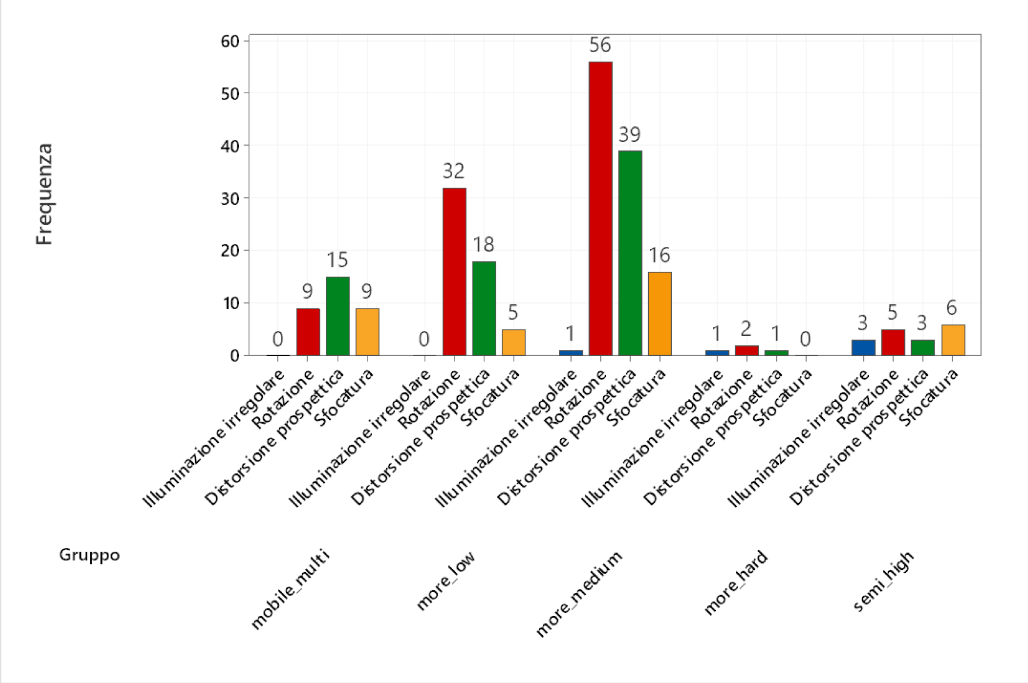


Figura 5.15: Frequenza delle tipologie di distorsione presenti nel dataset 3 nella parte con QR Code multipli

27 simboli. Perciò viene adottata la modalità di codifica byte e il numero di moduli richiesti, in tutti e tre i casi di dati codificati, comporta la generazione di QR Code di versione 3-L.

Le immagini nel dataset vengono distribuite in 5 gruppi *mobile_multi* (9), *more_low* (11), *more_medium* (22), *more_hard* (2) e *semi_high* (8): essi si differenziano nel dispositivo di acquisizione della foto, nella la tipologia di distorsioni introdotte e, a differenza dei dataset 1 e 2, nell'ambiente in cui è stato inserito il QR Code nell'immagine.

I gruppi *mobile_multi* e *more_hard* sono composti da immagini scattate dallo smartphone HTC, mentre i rimanenti dalla fotocamera Canon. In generale le distorsioni presenti nei diversi gruppi comprendono prevalentemente la rotazione, la deformazione prospettica e la sfocatura, in pochissimi casi anche un'illuminazione irregolare: esse sono applicate con diversi livelli di intensità in ogni singolo caso. Più nello specifico si evidenzia che i gruppi *mobile_multi*, *more_low* e *semi_high* presentano QR Code inseriti in un ambiente più sgombro da oggetti, in cui i codici risultano il soggetto principale e quindi facilmente localizzabili. Tuttavia, mentre i primi due gruppi ritraggono i codici in un'ambientazione in spazi interni con illuminazione omogenea, l'ultimo gruppo presenta uno sfondo che ritrae spazi esterni, dove sono presenti giochi di luce che creano un'illuminazione irregolare. I rimanenti gruppi, ovvero *more_medium* e *more_hard*, contengono immagini dove i QR Code si trovano spesso in secondo piano e occupano un'area molto ridotta, disposti su uno sfondo più rumoroso affollato da molti oggetti e, in alcuni casi, da simboli ad-hoc simili all'arrangiamento dei moduli nei codici per confondere il decodificatore.

Inoltre, i primi tre gruppi sopracitati sono caratterizzati da una media di 2-3 QR Code presenti per singola immagine, mentre negli ultimi due gruppi la media è più elevata, di circa 4-5 QR Code, con massimi di 14 codici. In generale la compresenza di QR Code è organizzata in tre modi diversi: nel primo approccio i QR Code si trovano tutti nello stesso piano e con le stesse dimensioni, nel secondo approccio i QR Code sono disposti sullo stesso piano ma con dimensioni differenti e nell'ultimo approccio i QR Code sono presenti su piani diversi e con dimensioni diverse. Questo ultimo approccio è diffuso principalmente nel gruppo *mobile_multi*, gli altri due approcci caratterizzano equamente le immagini nei rimanenti gruppi.

La tipologia di distorsione varia anche tra QR Code presenti nella stessa immagine: per esempio capita che i QR Code in primo piano sono messi a fuoco, mentre quelli in secondo piano risultano sfuocati. Alcuni esempi di immagini sono riportati in Fig. 5.13.

In Fig. 5.15 è illustrato un grafico che rappresenta, per ogni gruppo, la frequenza di ciascuna tipologia di distorsione presente nei QR Code contenuti nelle immagini della porzione con QR Code multipli del dataset 3, si tenga in considerazione che possono essere applicate in

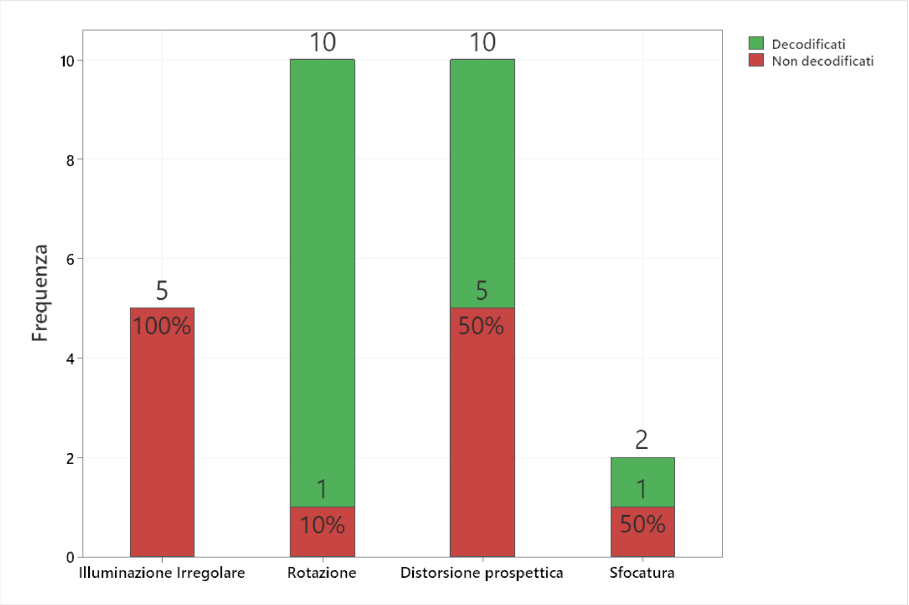


Figura 5.16: Frequenza di ogni tipologia di distorsione nei QR Code non decodificati, rapportata percentualmente sulla frequenza totale dei casi della relativa tipologia di distorsione, nel gruppo *one_medium* nella porzione con QR Code singoli del dataset 3

contemporanea più distorsioni.

5.3.2 Risultati

Le immagini vengono trasformate in matrici di valori attraverso opencv, allo stesso modo descritto in Sezione 5.1, e poi vengono elaborate dal decodificatore di pyzbar.

Parte con QR Code singoli

I risultati del test, confrontati con i dati originariamente codificati (riportati nel file csv), mostrano una corretta decodifica in 24 immagini su 29 totali, corrispondente a un tasso del 83%. Analizzando i 5 casi in cui non è avvenuta la decodifica, che appartengono tutti al gruppo *one_medium*, emerge che l'elemento comune tra le immagini è l'illuminazione irregolare (Fig. 5.16), assente nei casi correttamente codificati: le condizioni luminose anomale con cui è stata scattata la foto comportano troppi errori in fase di thresholding, rendendo impossibile la decodifica dei dati contenuti nel QR Code. La stessa problematica ha influito pesantemente anche nel dataset 1.

Questo evidenzia che quasi la totalità dei QR Code è stata robusta al rumore delle distorsioni, specialmente per i livelli presenti di rotazione, deformazione prospettica e sfocatura. Neanche la presenza di uno sfondo più rumoroso, introdotto per confondere il decodificatore, ha influito sulla corretta localizzazione dei codici.

Parte con QR Code multipli

Nella seguente analisi dei risultati, confrontati con i dati originariamente codificati (riportati nel file csv), si procede focalizzandosi singolarmente su ogni gruppo, dato che le immagini non sono numerose, in questo modo vengono colte in modo più mirato le peculiarità dei casi non decodificati. In Fig. 5.17 sono riportate le statistiche relative alle tipologie di distorsione nei QR Code non decodificati.

Il gruppo *mobile_multi* mostra una corretta decodifica in 15 QR Code su 22 totali in 9 immagini, pari al 68% dei casi. In particolare, analizzando i casi non decodificati, emergono 4 immagini in cui tra tutti i QR Code presenti uno è sfuocato e questo ne ha impedito la corretta decodifica. Nelle rimanenti immagini il problema comune è la presenza contemporanea di QR Code su piani differenti e con giaciture diverse. Molto probabilmente questo impatta sulla fase di demodulazione, dato che, in primo luogo, la nitidezza varia tra il codice in primo piano e l'altro in secondo piano per via delle condizioni di messa a fuoco, e poi interviene la deformazione a causa delle orientazioni diverse dei piani. In secondo luogo, essendo le foto acquisite in ambienti interni con illuminazione artificiale, hanno contribuito anche le condizioni differenti di luminosità tra i diversi piani, in base alla distanza e all'orientazione rispetto alla sorgente luminosa.

Il gruppo *more_low* mostra una corretta decodifica in 34 QR Code su 37 totali in 11 immagini, pari al 92% dei casi. Più nello specifico solo un'immagine contenente 3 QR Code non è stata decodificata: i 3 codici presenti sono interessati da una forte distorsione prospettica e una considerevole sfocatura. Nei rimanenti casi il livello di intensità delle distorsioni si è mantenuto basso, coinvolgendo solamente la rotazione e la sfocatura. Inoltre, a differenza dal gruppo precedente, i QR Code all'interno di ogni immagine si trovano tutti sullo stesso piano, sebbene abbiano dimensioni diverse, e presentano condizioni di luminosità omogenee.

Il gruppo *more_medium* mostra una corretta decodifica in 86 QR Code su 100 totali in 22 immagini, pari al 86% dei casi. Nel dettaglio in 2 immagini tutti i QR Code presenti non sono state decodificati: nel primo caso è presente un contrasto tra la porzione in ombra e la porzione illuminata che porta a una condizione di luce irregolare problematica per il thresholding, nel secondo caso tutti i codici sono sottoposti a una forte sfocatura e a distorsione prospettica. Gli altri QR Code non decodificati si trovano in immagini insieme ad almeno un QR Code decodificato con successo, ma i primi presentano importanti sfocature che impediscono la corretta demodulazione. Come il gruppo precedente, i QR Code hanno dimensioni diverse ma si trovano tutti sullo stesso piano all'interno di ogni immagine.

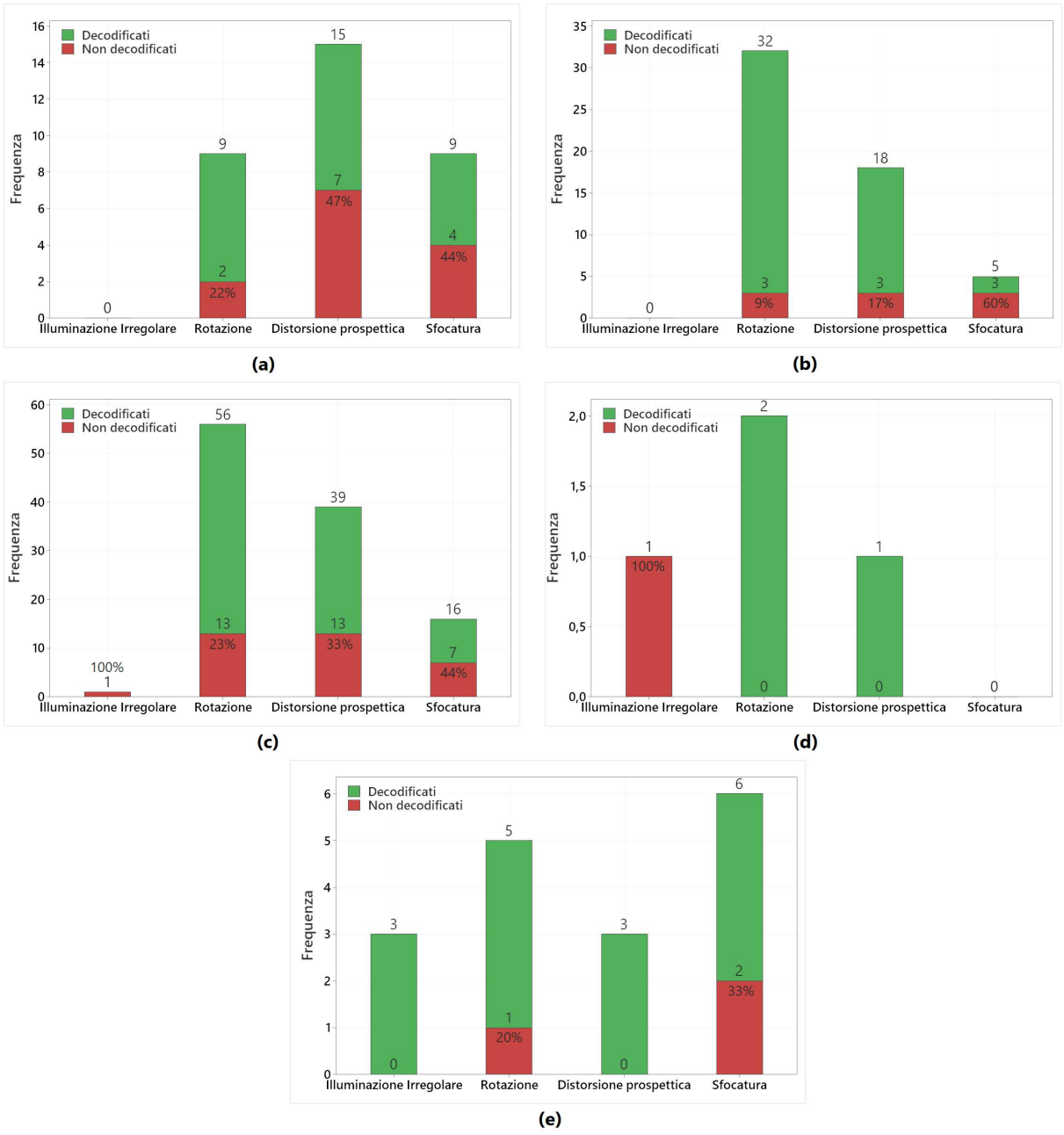


Figura 5.17: Frequenza di ogni tipologia di distorsione nei QR Code non decodificati, rapportata percentualmente sulla frequenza totale dei casi della relativa tipologia di distorsione, nei gruppi *mobile_multi* (a), *more_low* (b), *more_medium* (c), *more_hard* (d) e *semi_high* (e) contenuti nella porzione con QR Code multipli del dataset 3

Il gruppo *more_hard* mostra una corretta decodifica in 3 QR Code su 5 totali in 2 immagini, pari al 60% dei casi. Nel dettaglio questo gruppo contiene solo due immagini: la prima foto presenta 2 QR Code correttamente decodificati, la seconda foto presenta 3 QR Code di cui ne viene rilevato e decodificato solamente uno. In questo ultimo caso i due QR Code non sono decodificati a causa di un'illuminazione anomala e differente rispetto a quello decodificato, dato che un codice si trova in parte fortemente esposto alla luce e l'altro codice si trova in ombra, risultando insufficientemente illuminato.

L'ultimo gruppo *semi_high* mostra una corretta decodifica in 15 QR Code su 18 totali in 8 immagini, pari al 83% dei casi. Più nello specifico i 3 QR Code non decodificati, uno per immagine, presentano una risoluzione minore e una sfocatura maggiore rispetto agli altri QR Code presenti nell'ambiente catturato.

In conclusione, considerando tutti i gruppi insieme, si ottiene una tasso globale di decodifica corretta pari al 84%, in linea con i risultati ottenuti dall'altra porzione di dataset e dai precedenti dataset analizzati. Il decodificatore non ha fatto emergere particolari criticità nella localizzazione e nella decodifica di multipli QR Code in una stessa immagine, d'altro canto i problemi nella interpretazione delle immagini sono scaturiti a causa dell'elevata distorsione. In particolare, come negli altri dataset analizzati, l'illuminazione irregolare e la sfocatura sono le fonti di rumore più elevato per la demodulazione, comportando molti errori (superando la capacità correttiva dei codici RS) nel risultato prodotto dalla fase di thresholding. Inoltre, si aggiunge il contributo derivato dalla deformazione prospettica che impedisce un corretto dimensionamento del QR Code o dei moduli contenuti.

Conclusioni

Dall'analisi compiuta sui processi di codifica e decodifica dei QR Code e sullo Standard ISO/IEC 18004:2015 emerge la potenza di questi codici: immagazzinano fino a 7089 caratteri numerici e 4296 alfanumerici, grazie alla bidimensionalità e alle efficienti modalità di codifica (di sorgente) dei dati, e sono caratterizzati da una struttura progettata per permettere una scansione da tutte le angolazioni e senza allineamento, grazie ai pattern funzionali presenti. I codici di correzione degli errori Reed-Solomon, impiegati nella codifica di canale, raggiungono la massima distanza minima di Hamming fra le parole di codice, cioè la massima capacità correttiva. Lo stato dell'arte propone diverse tecniche per estendere la capacità dei QR Code da 2 fino a 24 volte superiore rispetto a quella dei QR Code standard. D'altro canto, nei casi in cui è presente una modulazione più complessa, questa comporta maggiori errori durante la decisione in fase di demodulazione e una maggiore sensibilità alle distorsioni introdotte dal canale, che riducono le prestazioni della decodifica. Nei casi di compressione il rapporto di compressione, quindi l'aumento di capacità del QR Code, varia in base al metodo scelto e all'ingresso fornito.

Dall'analisi dei dataset emerge che la struttura dei QR Code è discretamente robusta verso le distorsioni introdotte dal canale, dato che il tasso di decodifica varia tra 50% e 90%, con una media del 76%, in base al tipo e all'intensità delle distorsioni. Le tipologie di distorsione che hanno impedito maggiormente la decodifica dei QR Code sono l'illuminazione irregolare e la sfocatura: le condizioni di luce non omogenee impediscono di trovare una soglia opportuna per compiere il thresholding dell'immagine e stabilire correttamente il colore dei moduli, la sfocatura e l'assenza di nitidezza dell'immagine creano un forte rumore nella scansione, alterando i colori e i confini dei moduli, quindi in entrambi i casi viene compromessa la fase di demodulazione. Infine, gioca un ruolo importante anche la distorsione prospettica, soprattutto nei casi in cui il livello di intensità è elevato: essa sicuramente crea dei problemi di dimensionamento del QR Code per via delle deformazioni introdotte, causando quindi un'errata localizzazione dei pattern funzionali e delle posizioni dei moduli.

Negli sviluppi futuri di questa analisi sperimentale può essere indagato il processo di deco-

difica, anche con particolare riguardo alle prestazioni dei codici RS, in merito ad altre tipologie di distorsioni non analizzate in questo lavoro di tesi. Ad esempio si possono considerare QR Code danneggiati o incompleti, presi dalla realtà, oppure QR Code alterati artificialmente a livello digitale con particolari distorsioni ottiche (a barile, a cuscino, etc) o rumore (gaussiano, sale e pepe, etc.).

Bibliografia

[1] ISO/IEC 18004:2015. *Information technology — Automatic identification and data capture techniques — QR Code bar code symbology specification*. Standard. International Organization for Standardization, feb. 2015.

[2] Henryk Blasinski, Orhan Bulan e Gaurav Sharma. "Per-Colorant-Channel Color Barcodes for Mobile Applications: An Interference Cancellation Framework". In: *IEEE Transactions on Image Processing* 22.4 (2013), pp. 1498–1511. DOI: 10.1109/TIP.2012.2233483.

[3] Orhan Bulan e Gaurav Sharma. "High Capacity Color Barcodes: Per Channel Data Encoding via Orientation Modulation in Elliptical Dot Arrays". In: *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* 20 (nov. 2010), pp. 1337–50. DOI: 10.1109/TIP.2010.2092437.

[4] ComScore. *QR Code Usage Among European Smartphone Owners Doubles Over Past Year*. 19 Set. 2012. URL: https://www.comscore.com/Insights/Press-Releases/2012/9/QR-Code-Usage-Among-European-Smartphone-Owners-Doubles-Over-Past-Year (visitato il 05/07/2022).

[5] Herout A. Dubska M. e Havel J. *Real-time precise detection of regular grids and matrix codes*. URL: http://www.fit.vutbr.cz/research/groups/graph/pclines/pub_page.php?id=2012-JRTIP-MatrixCode (visitato il 08/08/2022).

[6] Hiren J. Galiyawala e Kinjal H. Pandya. "To increase data capacity of QR code using multiplexing with color coding: An example of embedding speech signal in QR code". In: *2014 Annual IEEE India Conference (INDICON)*. 2014, pp. 1–6. DOI: 10.1109/INDICON.2014.7030441.

[7] DENSO WAVE INCORPORATED. *QR Code development story*. URL: https://www.denso-wave.com/en/technology/vol1.html (visitato il 05/07/2022).

[8] DENSO WAVE INCORPORATED. *Retail Revolution of last 35 years*. URL: https://www.denso-wave.com/en/technology/vol2.html (visitato il 05/07/2022).

[9] Nicola Laurenti. "Channel Coding and Capacity". In: *Principles of Communications Networks and Systems*. John Wiley & Sons, Ltd, 2011. Cap. 6, pp. 373–429. ISBN: 9781119978589. DOI: https://doi.org/10.1002/9781119978589.ch6.

[10] Kinjal H Pandya e Hiren J Galiyawala. "A Survey on QR Codes: in context of Research and Application". In: *International Journal of Emerging Technology and Advanced Engineering* 4.3 (2014), pp. 258–262.

[11] Herout A. Szentandrasi I. e Havel J. *Fast detection and recognition of QR codes in high-resolution images*. URL: http://www.fit.vutbr.cz/research/groups/graph/pclines/pub_page.php?id=2012-SCCG-QRtiles (visitato il 18/08/2022).

[12] Sumit Tiwari. "An Introduction to QR Code Technology". In: *2016 International Conference on Information Technology (ICIT)*. 2016, pp. 39–44. DOI: 10.1109/ICIT.2016.021.

[13] Mona M. Umaria e G.B. Jethava. "A Novel Approach for Enhancing Data Storage Capacity in Quick Response Code Using Multiplexing and Data Compression Technique". In: *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*. 2015, pp. 1091–1093. DOI: 10.1109/CICN.2015.214.

[14] Max E. Vizcarra Melgar et al. "A High Density Colored 2D-Barcode: CQR Code-9". In: *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. 2016, pp. 329–334. DOI: 10.1109/SIBGRAPI.2016.052.

[15] Max E. Vizcarra Melgar et al. "CQR codes: Colored quick-response codes". In: *2012 IEEE Second International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*. 2012, pp. 321–325. DOI: 10.1109/ICCE-Berlin.2012.6336526.

[16] Sartid Vongpradhip. "Use multiplexing to increase information in QR code". In: *2013 8th International Conference on Computer Science & Education*. 2013, pp. 361–364. DOI: 10.1109/ICCSE.2013.6553938.

[17] Kim Ho Yeap et al. "A simple data storage system using QR code". In: *2014 5th International Conference on Intelligent and Advanced Systems (ICIAS)*. 2014, pp. 1–5. DOI: 10.1109/ICIAS.2014.6869536.