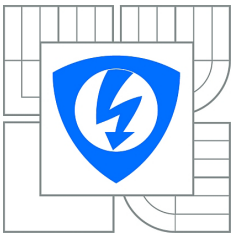
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

SYSTÉM PRO ROZPOZNÁVÁNÍ ČÁROVÝCH KÓDŮ
BARCODE RECOGNITION SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

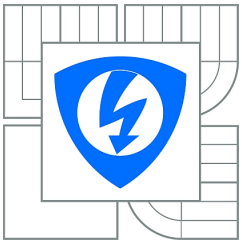
AUTOR PRÁCE
AUTHOR

Bc. WOJCIECH PRIBULA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR PETYOVSKÝ

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Wojciech Pribula	***ID:***	136577
Ročník:	2	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Systém pro rozpoznávání čárových kódů

POKYNY PRO VYPRACOVÁNÍ:

Cílem je navrhnout systém včetně algoritmů pro rozpoznávání čárových kódů používaných pro doručování zásilek.
1. Prostudujte problematiku čárových kódů.
2. Seznamte se s postupy a metodami detekce čárových kódů používanými pro doručování zásilek.
3. Na základě nastudovaných znalostí zvolte dva vhodné typy uspořádání čárových kódů, případně navrhněte vlastní.
4. Navrhněte vhodné uspořádání optické scény pro snímání čárových kódů pomocí CCD kamery. Vytvořte množinu testovacích snímků pro zvolené typy čárových kódů.
5. Navrhněte a realizujte algoritmy (ve formě OpenCV rozšíření) pro optickou detekci a rozpoznání čárových kódů.
6. Určete přesnost klasifikace čárových kódů na základě vyhodnocení souboru testovacích snímků.
7. Zhodnoťte dosažené výsledky a navrhněte další možná rozšíření.

DOPORUČENÁ LITERATURA:

[1] SONKA,M. , HLAVAC, V. , BOYLE, R.: Image Processing, Analysis, and Machine Vision, 3rd Edition, Thomson 2007, ISBN 049508252X.
[2] ŽÁRA, J.; BENEŠ, B.; FELKEL, P. : Moderní počítačová grafika, Computer press, 1998, ISBN 80-7226-049-9.

Termín zadání:	9.2.2015	***Termín odevzdání:***	18.5.2015

Vedoucí práce: Ing. Petr Petyovský
Konzultanti diplomové práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

Abstrakt
Tato práce popisuje čárové kódy používané pro poštovní služby. Konkrétně se jedná o Intelligent Mail Barcode, kód GS1-128, kód České pošty C128 a QR-kód. Jsou zde rozebrány způsoby zakódování informace do podoby kódu a příslušné opravné algoritmy používané pro opravu chyb při dekódování. Zvláště je věnována pozornost Reed-Solomonově korekci chyby u QR-kódu. Dále jsou probrány a zhodnoceny různé metody nalezení kódů, které jsou navrhované autory odborných článků. Je také popsán způsob tvorby testovacích sad snímků a navrhovaný vzhled snímací scény. V poslední řadě je zde umístěn popis vytvořených algoritmů pro detekci a dekódování čárových kódu GS1-128, C128 a IMB ve snímku a vyhodnocení úspěšnosti detekce navržených algoritmů.

Summary
This thesis describes barcodes which are used in postal services. Specifically, it is concerned with Intelligent Mail Barcode, the GS1-128 code, the C128 code of Ceska posta (Czech Postal Services) and the QR code. The thesis attempts to analyze methods of encoding information into barcodes and error detection algorithms used for error correction during the decoding processes. Most importantly, there is described Reed-Solomon error correction in the QR code. There are presented and evaluated different methods of code detecting which are suggested by authors of various academic articles. The thesis also describes the method of creating test sets of images and proposed appearances of the scanning scene. Additionally, there are described algorithms for detection and decoding barcodes GS1-128, C128 and IMB in the image which was created during the work on this thesis. Finally, there is the evaluation of the percentage success of algorithms.

Klíčová slova
čárový kód, čtení čárového kódu, poštovní čárové kódy, zpracování obrazu, rozpoznání vzoru, intelligent mail barcode, IMB, GS1-128, C128, QR-kód, Reed-Solomon, Galoisovo těleso, korekce chyb

Keywords
barcode, barcode reading, postal barcodes, image processing, pattern recognition, intelligent Mail Barcode, IMB, GS1-128, C128, QR code, Reed-Solomon, Galois field, error correction

PRIBULA, W.*Systém pro rozpoznávání čárových kódů*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2015. 86 s. Vedoucí Ing. Petr Petyovský.

Prohlašuji, že svou diplomovou práci na téma Sytém pro rozpoznávání čárových kódů jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Bc. Wojciech Pribula

Děkuji mému vedoucímu práce Ing. Petru Petyovskému za cenné rady ohledně zvolené problematiky a za neskutečnou pomoc při hledání potřebných materiálů.

Bc. Wojciech Pribula

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

1D a 2D kódy neboli čárové a maticové kódy neodmyslitelně patří k našim životům. Identifikujeme jimi nejen výrobky, ale i knihy v knihovnách, členy různých skupin, a dokonce i pacienty v nemocnicích kvůli eliminování chyb při podávání léků nebo provedení špatného zákroku.

Svůj náramek s kódem dostanou i čerstvě narozené děti, proto se dá klidně říct, že jsou kódy s námi už od narození. Lidstvo si na ně zvyklo a ani je nevnímá, pokud je nepotřebuje. Kolikrát se stane, že člověk ani neví, že na stránce časopisu je QR-kód, že i na láhvi piva, které si dopřává na konci těžkého dne, je přítomno číslo zakódované do bílých a černých pruhů.

Původní kódy EAN-13, které byl člověk ještě schopný víceméně dekódovat a přečíst sám, bez pomoci stroje, byly dnes doplněny sofistikovanějšími variantami. Dnes se data pro kód upravují a přidává se k nim množství kontrolních a samoopravných prvků. Některé kódy dokonce nejsou vidět, jsou natištěné barvou lidskému oku neviditelnou, ale správně vybavené zařízení je schopno kód přečíst lépe, protože si ho lehce odseparuje od pozadí.

V robotice se kódy používají pro snadnou orientaci stroje v prostoru. Bylo jimi nahrazeno velmi složité rozpoznávání reálných objektů tam, kde je to možné, a tím se celá oblast posunula o velký krok vpřed.

V dnešní době není problémem pořízení levného fotoaparátu nebo kamery, ale je problém pořídit si levné zařízení, které je schopno číst větší množství různých 1D a 2D kódů víceméně samostatně. Proto by bylo vhodné navrhnout sadu funkcí, které budou schopny lokalizovat, přečíst a dekódovat 1D a 2D kódy a budou univerzálně použitelné na různých zařízeních ať už mobilních, nebo stacionárních. Výhodou by samozřejmě byl open-source charakter řešení. Proto bylo rozhodnuto vytvořit funkce pro knihovnu OpenCV, které budou schopny zpracovat snímky s 1D a 2D kódy.

Tato práce se konkrétně zabývá kódy používanými na zásilkách, a to balíčcích nebo psaních, protože se jedná o oblast, kde je nejčastěji zapotřebí vypořádat se s více druhy kódů, například pro určení adresáta, určení údajů pro celní úřad nebo dopravce.

Na dalších stránkách budou postupně popsány kód GS1-128 pro popis vlastností balíků, kód IMB (*Intelligent Mail Barcode*) pro identifikaci a strojové třídění dopisů a QR-kód používaný obecně pro uložení jakýchkoliv dat. Budou zde také zmíněny opravné algoritmy a ochrany proti špatnému čtení, které jsou ve větší či menší míře implementovány ve všech druzích kódů.

Zvláště obtížné je pochopení způsobu oprav u QR-kódu pomocí Reed-Solomonovy korekce chyby. Způsob výpočtu korekce bude proto podrobněji probrán v samostatné kapitole, která se bude snažit nastínit postup, který je možné dále využít při tvorbě algoritmu.

Následovat bude popis současných metod lokalizace kódů, které jsou popsány ve vědeckých článcích v různých časopisech. Metody budou popsány, rozebrány a bude zhodnocena jejich případná použitelnost.

Bude také uveden návrh snímací scény pro případné použití ve funkci snímacího zařízení v provozu, které bude schopné číst různé typy kódů.

Součástí práce je také tvorba sady testovacích snímků, která bude sloužit k vývoji a testování algoritmů. Budou vytvořeny testovací zásilky s potiskem a polepem kódy, které budou snímány různými zařízeními.

Dále je zařazena kapitola, která se věnuje vývoji algoritmů, které budou schopny najít kód ve snímku, převést ho na binární data a správně ho dekódovat. Budou zde popsány fungující postupy, ale i ty, které byly vyzkoušeny, ale zklamaly a nakonec nebyly použity. Někdy je důležité pochopit, jak vypadal vývoj nějaké části algoritmu od prvotní základní myšlenky přes její modifikace až ke konečné verzi. Toto umožňuje pochopit, proč je daná část udělána takto, a ne jinak, a jak se vůbec dospělo k takovému, a ne jinému řešení.

V poslední řadě zde nesmí chybět kapitola věnována testům algoritmů na snímcích ze dvou dostupných snímacích zařízení, zrcadlovky Canon EOS 500D a mobilního telefonu Nokia 620. Tato zařízení tak rozdílná kvalitou i způsobem fotografování umožní porovnat úspěšnost algoritmů na kvalitativně úplně odlišných snímcích.

2. Popis čárových kódů

Tato kapitola se věnuje popisu čárových kódů. Postupně budou probrány kódy používané na zásilkách.

Bude popsán způsob úpravy informací a dat pro zakódování. Dále způsob převodu na kód a úprava kódu.

Informace jsou jen shrnutím a často obsahují odkazy na dokumentace na příslušné převodní tabulky, které zde nejsou umístěny z důvodu své obsáhlosti a lehké dostupnosti v dokumentaci.

2.1. Kód GS1-128 používaný pro popis vlastností balíku.

EAN - European Article Number (*cs.: Evropské číslo produktu*). Jiné názvy tohoto kódu jsou EAN-128 nebo UCC.

Jedná se o jednu z variant kódu GS1, kterou na svých balíčcích používá mezi jinými i společnost Amazon, který je jedním z největších rozesílatelů balíků na světě [25]. Speciálně varianta UCC/EAN-128 slouží pro označení zboží a umožňuje zapsání mnoha informací, jako datum výroby, datum spotřeby, hmotnost, velikost atd.

Výhodou oproti čárovému kódu používanému běžně v obchodech pro identifikaci zboží (EAN-13) je možnost kódování nejen číslic, ale i písmen a speciálních znaků.

2.1.1. Struktura kódu GS1-128

Kód se skládá z identifikátoru, který značí, co kóduje část za ním (seznam identifikátorů je standardizován a uveden ve specifikaci [9]). Dále jsou zde samotné informace/data a kontrolní součet.

Aplikační identifikátor	Data	Kontrolní číslo
$X_{1..2}$, $X_{1..3}$, $X_{1..4}$	$N_{1..n-1}$	N_n

Tabulka 2.1: Vzhled UCC/EAN-128

Výhodou kódu je, že se můžou za sebe řetězit až čtyři různé datové úseky. Jednotlivé úseky dat jsou odděleny znaky FNC1, FNC2, FNC3 a FNC4.

Komplikovanější je čtení identifikátoru, který je v délce dva až čtyři znaky. Na toto standard myslí a z počátečních číslic identifikátoru lze zjistit jeho délku. Například identifikátory začínající číslicí 9 jsou délky dvou znaků. Zato identifikátory začínající číslicí 7 jsou délky čtyři nebo tři znaky, ale identifikátory se čtyřmi znaky jsou formátu $700x$ a se třemi $71x$, kde x je jiná číslice. Vše je vyřešeno tak, aby nedocházelo ke konfliktům.

Příklad kódu GS1-128

Příklad štítku s kódem GS1-128 je na obrázku 2.1. Z dokumentace lze vyčíst, že identifikátor 02 znamená následnost GTIN - Global Trade Item Number (*cs.: Globální obchodní číslo produktu*), který je dlouhý 13 číslic plus jedna kontrolní, což odpovídá stavu na obrázku, a kóduje unikátní číslo produktu.

Následuje identifikátor 15, který odpovídá Best Before Date (*cs.: Datum trvanlivosti*), co značí datum, do kterého bude mít produkt specifikované vlastnosti (nejedná se jen

o datum spotřeby na živočišných produktech, ale i o jiné produkty s předpokládaným stárnutím). Z dokumentace lze vyčíst formát čísla za identifikátorem (2.2), ale není nutná implementace dokumentace pro čtení kódu, jelikož je datový řetězec ukončen stop znakem nebo znakem označujícím začátek dalšího datového řetězce.

Identifikátor	Rok	Měsíc	Den
1 5	N_1 N_2	N_3 N_4	N_5 N_6

Tabulka 2.2: Formát kódu s identifikátorem 15.

Dále následuje datový řetězec s informacemi, které výrobce považuje za relevantní.

Poslední je množství zboží s identifikátorem 37.

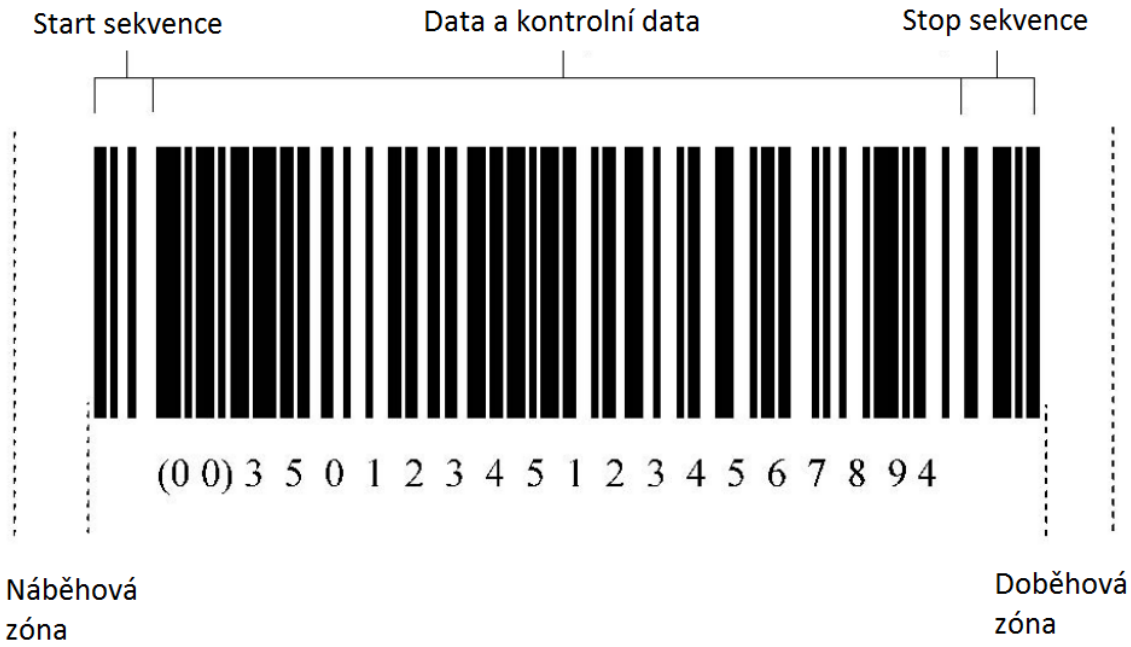
Druhý čárový kód s identifikátorem 00 je SSCC - Serial Shipping Container Code (*cs.: Sériový nákladní kontejnerový kód*). Jedná se o unikátní kód zásilky.

Obrázek 2.1: Příklad štítku s kódem UCC/EAN-128. [27]

Způsob kódování GS1-128

Vzhled čárového kódu je patrný z 2.2. Čtení se provádí zleva doprava a kód obsahuje:

- náběhovou zónu,
- start sekvenci,
- data, ověřovací data a speciální znaky,
- stop sekvenci,
- doběhovou zónu.



Obrázek 2.2: Struktura GS1-128 kódu. [9]

Start sekvence

Start sekvence je dlouhá 11 základních šířek kódu (bitů). Ve start sekvenci je zakódováno, jak se budou další znaky interpretovat. Existují tři varianty. Varianta **A**, **B** a **C** (2.3).

Varianta A - obsahuje velká písmena, číslice 0-9, speciální znaky, zkratky a další speciální sekvence.

Varianta B - obsahuje malá i velká písmena abecedy, číslice 0-9 a speciální znaky.

Varianta C - tato varianta kóduje čísla 00-99.

Kromě toho každá varianta obsahuje znaky přechodu na každou ze zbylých variant.

Varianta	1	2	3	4	5	6	7	8	9	10	11
A	■	■		■					■		
B	■	■		■			■				
C	■	■		■			■	■	■		

Tabulka 2.3: Start sekvence. [9]

Data

Znaky se kódují sekvencí o délce jedenácti základních šířek (bitů) s tím, že poslední je vždy bílý a první vždy černý. Takže v důsledku jsou znaky kódovány 9 bity. Čáry jsou vždy tři černé a tři bílé.

V závislosti na kódování se v kódu můžou vyskytovat písmena velká i malá, speciální znaky, čísla, zkratky, příkazy. Jsou vyhrazeny i znaky pro změnu varianty. Takže může třeba dojít ke změně z kódovací varianty A na variantu C v polovině kódu.

Stop sekvence

Stop sekvence je dlouhá 13 bitů a je vždy stejná (2.4).

1	2	3	4	5	6	7	8	9	10	11	12	13
■	■				■	■	■		■		■	■

Tabulka 2.4: Stop sekvence. [9]

Stavba kódu

Kód se uvede start sekvencí. Následuje znak FCN1 pro počátek prvního řetězce. Potom, pokud je potřeba, následuje změna na variantu C a je uveden identifikátor. Následují data nebo další změny variant. Po datech je uveden kontrolní znak. Dále je buďto uveden znak FNC2 a následuje další identifikátor a data, nebo je vše ukončeno stop sekvencí. Řetězit se můžou až čtyři datové řetězce.

2.1.2. Výpočet kontrolního čísla

Algoritmus výpočtu je stejný pro všechny délky.

Jedná se pouze o nejrozšířenější výpočet doporučený standardem GS1, ale může se použít jakýkoliv jiný výpočet, který ovšem musí být znám čtecímu zařízení.

1. Očíslují se pozice zleva doprava od 1 do n. Liché pozice se vynásobí 3.
2. Všechna čísla se sečtou.
3. Kontrolní číslo je hodnota chybějící do nejbližšího, vyššího násobku 10. Například pro 101 je nejblíže 110, takže kontrolní číslo je 9.

2.2. Kód C128 České pošty pro označení doporučených zásilek

Tento čárový kód vychází z čárového kódu UCC/EAN-128, který byl popsán výše. Česká pošta pouze specifikuje vzhled kódu a obsažené informace. Informace čerpány z [28].

2.2.1. Rozměry kódu C128

Způsob tisku kódu je na obrázku 2.3.

Modulová šířka - 0.25 - 0.375 mm

Délka čárového kódu - 40 - 69 mm bez ohranných zón

Ochranné zóny - doporučuje se minimálně 9 mm

Výška - optimální je 2/3 - 3/3 délky čárového kódu, minimálně 10 mm

Další parametry nebudou na tomto místě zmíněny, ale lze je dohledat ve specifikaci [28].

2.2.2. Identifikační číslo doporučeného psaní

Číslo pro identifikaci zásilky.

Zásilkové firmy

Firmy s podáním přes 1 000 000 ks zásilek ročně.

TT	DD	IIIIIII	K	X
2	2	7	1	1

Tabulka 2.5: Struktura identifikačního čísla pro zásilkové firmy u České pošty. [28]

TT - označení druhu zásilky

DD - číslo podavatele (1-99)

IIIIIII - podací číslo zásilky

K - kontrolní číslice

X - identifikace typu podavatele (F - zásilková firma)

Sběrná pošta - velký podavatel

Podavatelé s podáním do 1 000 000 ks zásilek ročně.

TT	PP	D	IIIIII	K	X
2	2	1	6	1	1

Tabulka 2.6: Struktura identifikačního čísla pro sběrné pošty - velcí podavatelé. [28]

TT - označení druhu zásilky

PP - číslo sběrné pošty (1-99)

D - číslo podavatele na sběrné poště (1-9)

IIIIII - podací číslo zásilky

K - kontrolní číslice

X - identifikace typu podavatele (U - velký podavatel na sběrné poště)

Obrázek 2.3: Parametry pro podací nálepku na Doporučené zásilky. Rozměry jsou udány v milimetrech. [28]

Sběrná pošta - středně velký podavatel

Podavatelé s podáním do 100 000 ks zásilek ročně.

TT	PP	DD	IIIII	K	X
2	2	2	5	1	1

Tabulka 2.7: Struktura identifikačního čísla pro sběrné pošty - středně velcí podavatelé. [28]

TT - označení druhu zásilky

PP - číslo Sběrné pošty (1-99)

DD - číslo podavatele na sběrné poště (1-99)

IIIII - podací číslo zásilky

K - kontrolní číslice

X - identifikace typu podavatele (C - středně velký podavatel na sběrné poště)

Sběrná pošta - malý podavatel

Podvatelé s podáním do 10 000 ks zásilek ročně.

TT	PP	DDD	IIII	K	X
2	2	3	4	1	1

Tabulka 2.8: Struktura identifikačního čísla pro sběrné pošty - malí podavatelé. [28]

TT - označení druhu zásilky

PP - číslo sběrné pošty(1-99)

DD - číslo podavatele na sběrné poště(1-999)

IIIII - podací číslo zásilky

K - kontrolní číslice

X - identifikace typu podavatele (M - malý podavatel na sběrné poště)

2.2.3. Kontrolní součet kódu C128

Podavatel může mít vlastní schválený algoritmus, ale obecně se používá modulo 11.

Všechny číslice numerické části identifikátoru se vynásobí odpovídajícím číslem a výsledky se sečtou.

A = zbytek po dělení $fracsum11$

$11 - A = K$

K...kontrolní číslo

D	D	I	I	I	I	I	I	I		
*	*	*	*	*	*	*	*	*		
1	8	6	4	2	3	5	9	7		
=	=	=	=	=	=	=	=	=		
s1	+s2	+s3	+s4	+s5	+s6	+s7	+s8	+s9	=	sum

Tabulka 2.9: Výpočet sumy pro metodu modulo 11.

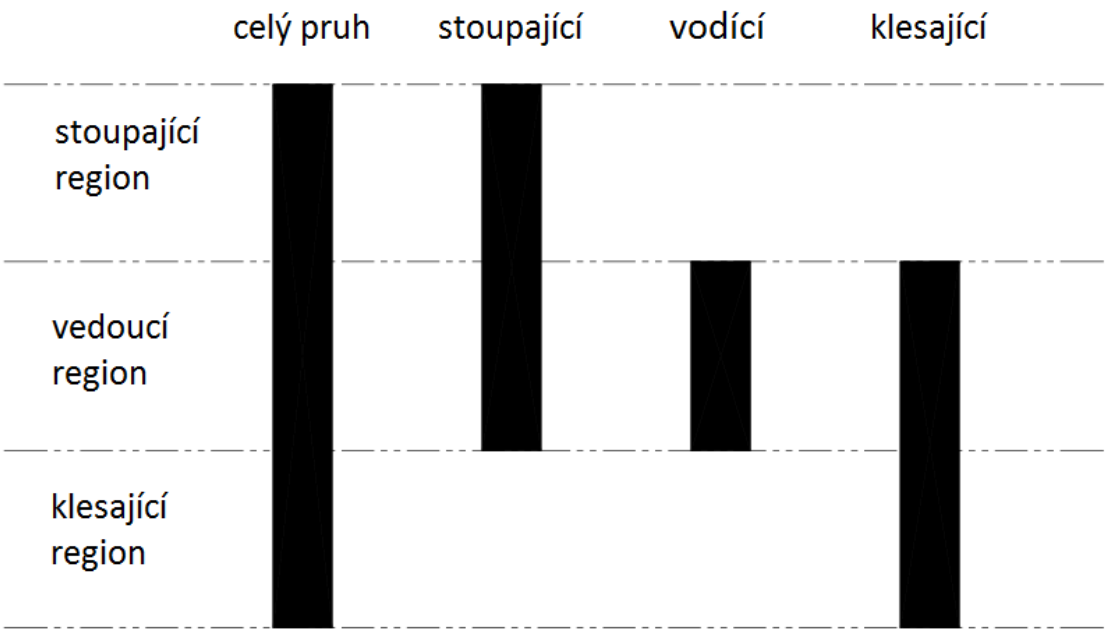
2.3. Inteligent Mail Barcode (IMB) pro označování obálek v třídicím stroji.

Informace jsou čerpány ze specifikace Poštovního úřadu Spojených států amerických [16].

Tento kód se jinak označuje jako USPS OneCode Solution nebo USPS 4-State Customer Barcode, popřípadě i zkratkou 4CB, 4-CB nebo OSPS4CB. Dále bude používáno označení IMB.

IMB byl vytvořen pro kódování směrovacích čísel a adres, hlavně u pohlednic a psaní. Tento kód je mezinárodní a je používán i Českou poštou. I když je řazen mezi 1D kódy, zařazení najde také v 2D kódech jelikož pro jeho přečtení nestačí vést jednu přímku přes kód.

Je to zapříčiněno tím, že kód používá čtyřkovou soustavu. Skládá se z proužků, které kódují čtyři stavy (Obrázek 2.4).



Obrázek 2.4: Čtyři stavy v IMB kódu. [16]

2.3.1. Složení IMB kódu

Kód se skládá z maximálně 31 znaků, co nakonec dává 65 pruhů. Obsažené informace znázorňuje tabulka 2.10.

Identifikátor kódu je označení, které specifikuje typ dat. Jestli se jedná o firmu, adresu distribučního centra, nebo jen specifikaci cesty.

Pole	**Znaky**
Identifikátor kódu	2 (druhý musí být 0-4)
Identifikátor typu služby	3
Idntifikátor odesílatele	6 nebo 9
Sériové číslo	9 (použití s 6místným ID schránky) 6 (použití s 9místným ID schránky)
Koncové poštovní směrovací číslo	0, 5, 9 nebo 11

Tabulka 2.10: Složení IMB kódu.

Identifikátor typu služby je z rozsahu 000-999. Lze určit, zda se jedná o běžnou zásilku, periodikum, obchodní zásilku atd.

Identifikátor odesílatele je unikátní číslo, které identifikuje podnikatelský subjekt. Šestimístný kód je z rozsahu 000000-899999 a devítimístný má povolený rozsah 900000000-999999999.

Sériové číslo by mělo být přiděleno odesílatelem pro možnost identifikace a sledování zásilky. Povoluje se rozsah 000000000-999999999 pro použití se šestimístným identifikátorem odesílatele a rozsah 000000-999999 při použití devítimístného identifikátoru odesílatele.

Koncové poštovní směrovací číslo kóduje koncové místo určení.

2.3.2. Proces kódování IMB

Proces kódování se skládá z šesti kroků.

1. Konverze na binární data

Konverze koncového poštovního směrovacího čísla

Záleží na délce:

0 míst - hodnota 0

5 míst - konverze na desetinné číslo formátu integer $+1_d$

9 míst - konverze na desetinné číslo formátu integer $+100\,000_d + 1_d$

11 míst - konverze na desetinné číslo formátu integer $+1\,000\,000\,000_d + 100\,000_d + 1_d$

Poté se provede konverze na binární data, s tím, že data, která se vešla do 37 bitů, by měla být v 37 bitech nejvíce vpravo.

Konverze zbylých dat. Všechno kromě PSČ se považuje za jedno číslo.

Vezmou se binární data z předešlé konverze PSČ. Vynásobí se desítkovou 10 a přičte se první číslice.

Poté se binární data vynásobí desítkovou 5 a přičte se druhá číslice.

Následně se vždy násobí desítkovou 10 a přičtou se postupně zbylé číslice.

2. Vygenerování 11bitového kontrolního součtu z binárních dat

Funkce generování kontrolního součtu je v uvedené literatuře [16]. Vstupem je pole se 102 bity. Dva bity nejvíce vlevo neobsahují data, proto musí být vynechány.

Jedná se o výpočet CRC (*cyclic redundancy check*). Je to dělení datového polynomu $D(x)$ generujícím polynomem $G(x)$. Polynomy jsou reprezentovány bity. Převod mezi polynomem a bity ilustruje následující příklad:

$$x^7 + x^5 + x^2 + x + 1 \rightarrow 1010\,0111_d \tag{2.1}$$

CRC je potom zbytek po dělení $\frac{D(x)}{G(x)}$ s tím, že se použije aritmetika Galiusova tělesa (ta je blíže vysvětlena v kapitole o QR-kódu 2.4). Ve zkratce se sčítání a odečítání vymění za jedinou operaci bitového XOR.

U IMB je generující polynom `0x0F35` v binární podobě `1111 0011 0101`. Pro jeho výpočet se doporučuje použít funkci z dokumentace, aby byl zachován přesný algoritmus výpočtu.

3. Konverze binárních dat na kódová slova

- Nejprve se vše vydělí 636, zbytek je kódové slovo J.
- Pak se dělí 1365, zbytek je kódové slovo I.
- Znova se dělí 1365, zbytek je kódové slovo H atd.
- Koncový zbytek by měl být v rozmezí 0-658 a je to kódové slovo A.

4. Přidání dodatečných informací do kódových slov

Kódové slovo J se převede z rozsahu 0-635 na sudá čísla z rozsahu 0-1270. Je to kvůli detekci převrácení kódu (bude vysvětleno).

Pokud je nejvýznamnější bit kontrolního součtu roven binární 1, přičte se ke kódovému slovu A 659.

5. Konverze z kódových slov na znaky

V závislosti na velikosti kódového slova se provede konverze na znaky podle tabulek uvedených v dokumentaci [16]. Dokumentace také uvádí funkce generující znaky, které se dají použít pro vygenerování tabulek za běhu programu a není nutné skladování obsáhlých tabulek.

Každý znak obsahuje 2 nebo 5 jedničkových bitů. [26]

Pro slova z rozsahu 0-1286 se použije první tabulka ze specifikace. U slov z rozsahu 1287-1364 se provede konverze: $slovo - 1287$ a použije se druhá tabulka ze specifikace.

Nyní už není řeč o slovech A-J, ale o znacích A - J.

Každý znak by měl být spárován s nepoužitými bity kontrolního součtu. Znak A s nejméně významným bitem, znak J s druhým nejvíce významným bitem.

Pokud je hodnota spárovaného bitu 1, provede se bitová negace znaku.

Tím pádem se budou vyskytovat jen znaky s 2, 5, 8 nebo 11 jedničkovými bity. [26]

6. Konverze ze znaků na Intelligent Mail Barcode

Na konci je obdržen kód o deseti slovech, každé 13 bitů dlouhé. To je 130 bitů celkem. Každý pruh může popsat dva bity, takže výsledný kód je 65 pruhů dlouhý.

Postupně se prochází řetězec bitů a pokud se na daném místě nachází bitová jednička, dohledá se toto místo v tabulce převodu na grafický kód a na určené místo se umístí klesající nebo stoupající pruh.

Tabulka je opět dostupná ve specifikaci [16].

Obrázek 2.5: Příklad natištěného IMB kódu. [16]

2.3.3. Detekce převrácení při čtení IMB

Tabulka konverze kódových slov na znaky použita v pátém kroku kódovacího procesu není poskládána náhodně. Znaky na konci tabulky jsou palindromy (čtou se stejně od počátku i od konce). Zbylé znaky, které nejsou palindromy, jsou poskládané tak, že znaky, které mají stejný zápis, ale zrcadlově otočený, jsou umístěny za sebou (Např. 0 - 0000000011111 a 1 - 1111100000000). Jedná se vždy o pár sudé a liché číslo.

Čtení zrcadleného znaku dá ve výsledku vždy stejné číslo u palindromu nebo jeden z párů u nepalindromu.

Zdánlivě náhodné rozložení znaků, které se děje v posledním kroku kódování, má svůj význam při detekci převrácení kódu. Bity jsou uloženy tak, že když se přečte kód převráceně, obrátí se pořadí bitů a číslo sudé se změní na liché číslo ze své dvojice, a jak bylo zmíněno dříve, slovo J je vždy sudé.

Jelikož se palindromy vyskytují až od slova 1272 a slovo J je maximálně 1270, je tato kontrola vždy platná.

2.3.4. Oprava chyb při čtení IMB

Oprava chyb funguje na základě nalezení znaků, které mají nesprávný počet jedničkových bitů. Lze takto detekovat chybu jednoho bitu ve znaku. Pro opravu se zkouší každý možný znak a ověřuje se CRC.

Je pravděpodobné, že když bude poškozen jeden bit ve znaku, bude poškozený celý pruh a tím pádem bude poškozený i jeden bit v jiném znaku (v jednom pruhu nejsou nikdy kódovány bity stejného znaku). Potom se zkouší všechny kombinace s těmito dvěma znaky. [26] Vždy se zneguje jeden bit znaku a dopočte se CRC, a když bude stejné jako to zakódované, tak i čtení by mělo být správné.

Dá se opravit i více chyb než dvě, ale náročnost výpočtu stoupá s každou chybou, proto se reálně opravují jen poruchy po jednom bitu ve dvou bytech.

Pokud počty jedničkových bitů odpovídají, ale CRC nesedí, jsou minimálně dvě chyby v jednom ze znaků. Lze hledat chybu hrubou silou, zkoušením negovat různé páry bitů

postupně v jednotlivých znacích. Existuje ale pravděpodobnost vícenásobné chyby, ve více slovech.

Může se vyskytnout i kombinace obou případů. V jednom znaku je chybný jeden bit a v jiném dva.

V případě, že by byl učiněn pokus opravit příliš velké množství bitů, tak se může stát, že se vygeneruje bitový proud, který bude mít správné CRC, ale chybně dekódovaná data. Je to způsobené tím, že CRC je zakódováno v proudu bitů.

Závěrem lze říci, že při práci s IMB kódem je potřeba dát si pozor na správné přetváření dat. Problémem je, že se s daty pracuje jako s jedním dlouhým číslem, které se nevejde do žádného běžného datového typu. Je také potřeba správně implementovat výpočet kontrolního součtu a jeho srovnání s hodnotami v kódu. Je vhodné implementovat algoritmus opravy chyb, který zkouší různé kombinace a hledá tu, která vede na schodu kontrolního součtu vypočteného a uloženého v kódu. Problémem je, že uložený kontrolní součet se v průběhu opravy chyby mění a může se stát, že nesprávná kombinace oprav povede na schodu kontrolních součtů.

2.4. QR-kód jako odkaz nebo datový kontejner pro popis zásilky

Poczta Polska s.a. používá na svých zásilkách čárový kód GS1-128 (kapitola 2.1). Tento kód je použit, jako v případě jiných poštovních institucí, pro zakódování údajů o zásilce.

Kromě tohoto kódu je přítomen i QR-kód, který slouží pro přesměrování na stránku Poczty Polské a konkrétně na službu sledování balíku a přehled informací o něm.

Takže zde tento kód není používaný ve smyslu kódování informací o balíku, ale jako doplňkový kód, který zajišťuje jiné služby kolem přepravy zásilek.

Na obrázku 2.6, který ukazuje vzor vyplnění formuláře pro odeslání zásilky, je viditelný jak kód GS1-128 s informacemi o zásilce, tak QR-kód, který umožňuje přesměrování na stránky se službou sledování balíku.

Toto je samozřejmě jen příklad a takto QR-kód používají i jiné firmy po celém světě.

Kód je dále také používán jako kontejner pro uchování informací o balíku a struktura dat potom připomíná strukturu dat u GS1-128 2.1.

POZNÁMKA: Následující popis kódu vychází ze specifikace [29].

2.4.1. Možnosti QR-kódu

QR-kód je 2D maticový kód schopný uchovávat různé znaky. Byl vyvinut tak, aby byl schopen zakódovat čísla, písmena, binární znaky a znaky Kanji (Obrázek: 2.8).

Zajímavé je množství uchovávaných dat, které samozřejmě závisí na verzi a velikosti matice kódu.

Nejmenší je Micro QR Code, Version M4-L (Tab. 2.11).

Největší je QR Code Version 40-L (Tab. 2.12)

Je zde patrné, že možnosti QR-kódu jsou dosti veliké co se týče množství uchovávaných dat. Další výhodou je možnost opravy chyb čtení pomocí Reed-Solomonovy korekce chyby. Jsou definovány čtyři úrovně dovolující obnovit definované procento dat:

numerická data	35 znaků
alfanumerická data	21 znaků
bytová data	15 znaků
Kanji data	9 znaků

Tabulka 2.11: Množství dat v Micro QR Code Vision M4-L.

numerická data	7 089 znaků
alfanumerická data	4 296 znaků
bytová data	2 953 znaků
Kanji data	1 817 znaků

Tabulka 2.12: Množství dat v QR Code Vision 40-L.

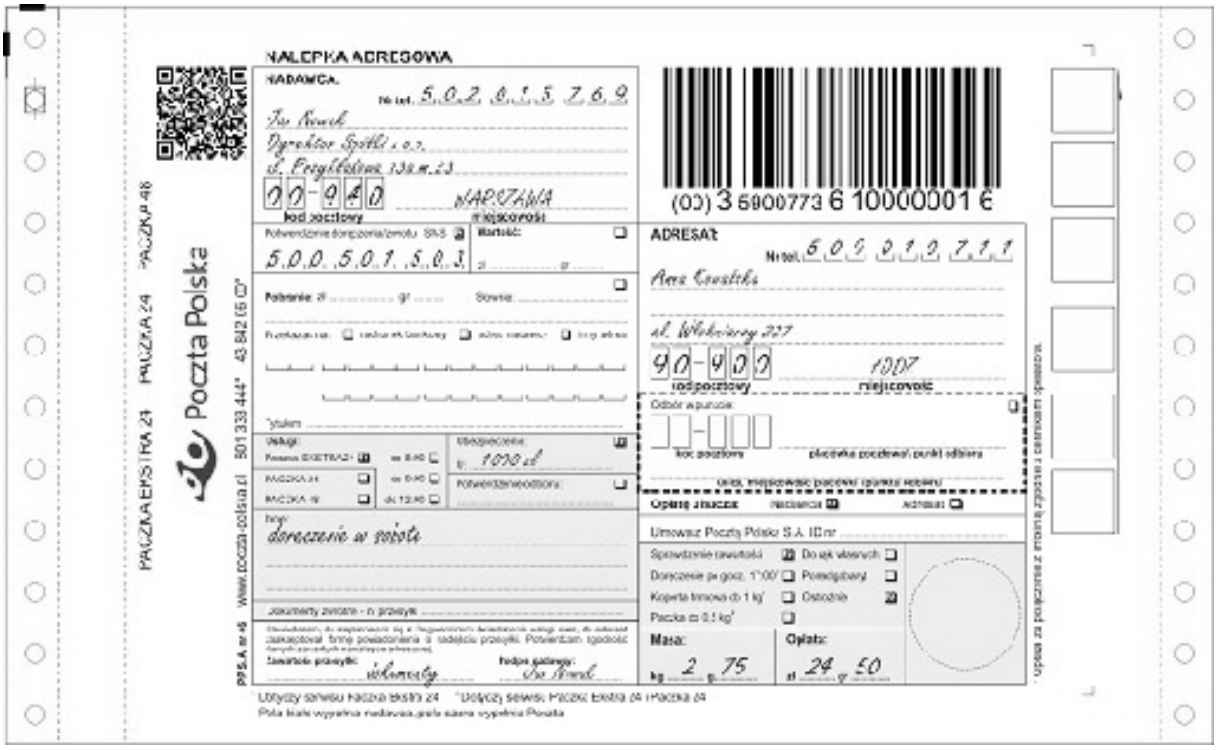
L	7%
M	15%
Q	25%
H	30%

Tabulka 2.13: Úrovně korekce chyb čtení QR kódu.

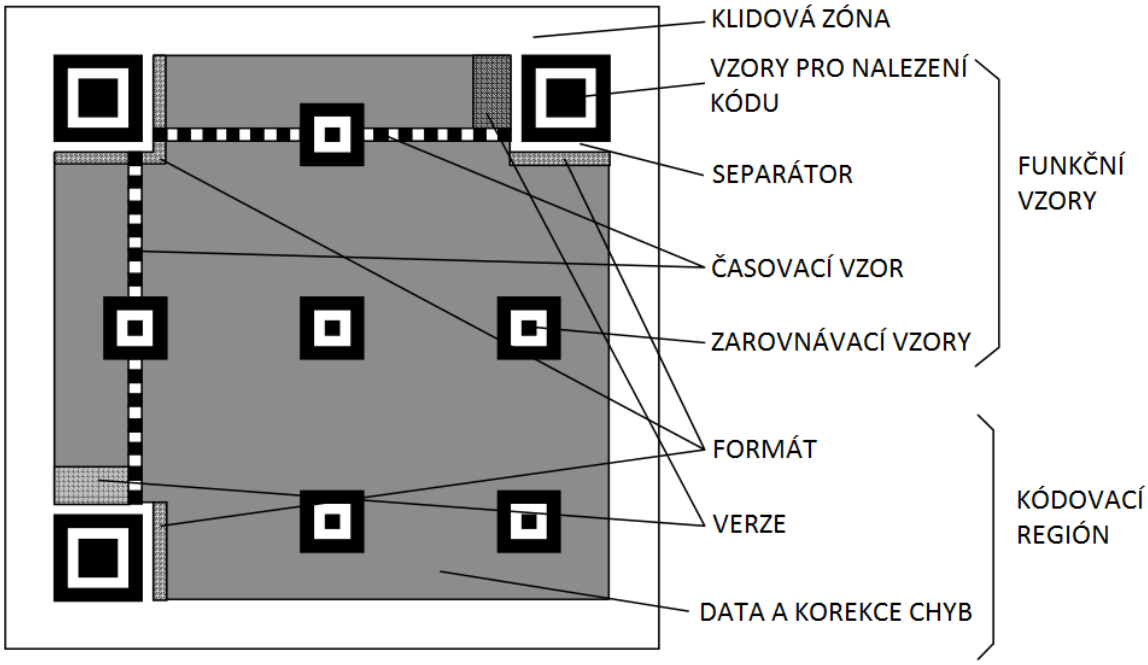
Pro malé verze jsou dostupné pouze některé úrovně nebo pouze detekce chyby bez její opravy.

2.4.2. Struktura dat v QR-kódu

Data jsou kódována světlými a tmavými čtverečky. Tmavé kódují binární 1_b a světlé binární 0_b.



Obrázek 2.6: Vzor vyplnění adresy s viditelnými kódy. [31]



Obrázek 2.7: Obecná struktura QR-kódu. [29]

Specifikace počítá i s reverzními barvami, ale veškerý další popis se bude týkat černého kódu na bílém pozadí.

Kód má být konstruován ze čtvercových modulů a měl by obsahovat region zakódovaných dat a funkční vzory.

Vše je znázorněno na obrázku 2.7 a bude dále blíže popsáno.

Funkční vzory

Vzor pro nalezení kódu - je to tmavý čtverec o šířce 8 modulů, v něm je světlý čtverec o šířce 6 modulů a v tomto čtverci je další černý čtverec se šířkou 4 moduly.

Separátory - separují vzory pro nalezení kódu a zakódovaná data. Jedná se o světlý pruh šířky jednoho modulu.

Časovací vzor - jedná se o dvě přímky složené ze světlých a bílých modulů, které se pravidelně střídají. Slouží ke kalibraci sloupců a řádků kódu.

Zarovnávací vzory - jejich množství je závislé na velikosti datové matice a slouží pro lepší zarovnání mřížky pro čtení dat. Jsou podobné vzorům pro nalezení matice. Skládají se z tmavého čtverce o šířce 5 modulů, v něm je světlý čtverec o šířce 3 modulů a uprostřed světlého čtverce je ještě čtverec o šířce 1 modulu.

Kódovací region

Formát - určuje formát dat zakódovaných v matici.

Verze - určuje verzi kódu a tím i jeho velikost a rozmístění funkčních vzorů.

Data a korekce chyby - samotná zakódovaná data a data potřebná pro korekci chyb.

Tichá zóna

Jedná se o světlou oblast kolem kódu, která by měla být široká 4 moduly pro QR-kód a 2 moduly pro mikro QR-kód.

Obrázek 2.8: Příklad znaků z abecedy Kandži. [30]

2.4.3. Módy kódování dat do QR-kódu

Je třeba rozhodnout, který mód kódování se použije pro data.

Extended Channel Interpretation (ECI) (*Rozšířená kanálová interpretace*) protokol dovolující rozdílnou než výchozí interpretaci znaků podle jejich typu. ECI protokol není podporován v Micro QR-kódu. Jedná se prakticky o implementaci různých znakových sad definovaných ISO normou. Například znaková sada ISO/IEC 8859-1 obsahující znaky latinské abecedy a dovolující zapsat slova povětšinou germánských a románských jazyků.

Numerický mód - kóduje čísla 0-9 (bitová hodnota 30_{HEX} - 39_{HEX}). Tři čísla jsou reprezentována 10 bity.

Alfanumerický mód - sada 45 znaků: 0-9 (bitová hodnota 30_{HEX} - 39_{HEX}), 26 písmen A-Z (bitová hodnota 41_{HEX} - $5A_{HEX}$) a 9 symbolů (SP, $, %, *, +, -, ., /, :)(bitová hodnota 20_{HEX}, 24_{HEX}, 25_{HEX}, $2A_{HEX}$, $2B_{HEX}$, $2D_{HEX}$, $2E_{HEX}$, $2F_{HEX}$, $3A_{HEX}$).

Tento mód není dostupný v M1 Mikro QR-kódu.

Bytový mód - data jsou kódována 8 bity na jeden znak. Kódovací tabulka se musí určit v kódu nebo oboustrannou dohodou.

Tento mód není dostupný v M2 a M1 Mikro QR-kódu.

Kandži mód - jedná se o znaky používané v Asii, převážně v Číně a Japonsku. Příklad znaků je na obrázku 2.8.

Používá se Shift JIS systém. Jedná se o obdobu ASCI tabulky pro abecedu Kandži. Znaky jsou kódovány 13 bitovými slovy. Slovo se skládá z dvou čísel určujících souřadnice v tabulce systému Shift JIS.

Tento mód je jeden z hlavních důvodů pro zavedení QR-kódů do praxe. Jelikož klasické 1D čárové kódy jsou postaveny na latinské abecedě a nejsou schopny optimálně řešit rozsáhlost asijských abeced.

Smíšený mód - tento mód říká, že QR-kód obsahuje více z předešlých módů, mezi kterými je přepínáno.

Strukturovaný přídavný mód - každý symbol obsahuje hlavičku, která blíže specifikuje jeho pozici a kódování.

FNC1 mód - mód, ve kterém jsou data kódována dle jiné specifikace. Vždy se týká celé matice.

2.4.4. Kódování dat do bitového proudu pro převod na QR-kód

ECI hlavička se skládá z:

- indikátor módu (4 bity),
- ECI označení (8, 16 nebo 24 bitů)

Zbytek bitového proudu tvoří:

- indikátor módu,
- indikátor počtu znaků,
- proud bitů dat,
- terminátor - konec zprávy.

Data lze rozdělit do segmentů, které začínají indikátorem módu a končí posledním bitem dat. Terminátor se umístí až za poslední segment.

ECI označení - Aby bylo možno rozlišit, jak dlouhé je ECI označení, rozlišuje se to počátečními bity. Vše je znázorněno v tabluce 2.14.

ECI přiřazené číslo	počet kódovacích slov	hodnota slov
000000 - 000127	1	0bbbbbbb
000000 - 016383	2	10bbbbbb bbbbbbbb
000000 - 999999	3	110bbbbb bbbbbbbb bbbbbbbb

Tabulka 2.14: Označení ECI. (b - binární hodnota přiřazeného čísla.)

Kódování dat v módech

Numerický mód - čísla jsou rozdělena do skupin po třech (poslední, kolik zbyde) a tyto skupiny jsou převedeny na svoje binární ekvivalenty.

Alfanumerický mód - každý znak obdrží hodnotu 0-44 podle tabulky. Znaky jsou rozděleny do skupin po dva znaky. Hodnota prvního znaku se vynásobí 45 a k výsledku se přičte hodnota druhého znaku. Poté se číslo převede na 11bitový binární ekvivalent. Pokud počet znaků není sudý, tak se poslední znak zakóduje 6bitovým binárním číslem.

Bytový mód - znaky jsou kódovány přímo osmibitovými slovy.

Kandži mód - tímto módem se tato práce nebude zabývat blíže. Na znaky Kandži bude nahlíženo jako na 13bitová čísla.

Smíšený mód - jak již bylo zmíněno dříve, v tomto módu lze přepínat mezi předešlými módy. Sestavení dat znázorňuje tabulka 2.15.

FNC1 mód v první pozici - navazuje na GS1-128 kód. Jelikož má velké uplatnění na zásilkách, bude uveden příklad použití (PŘÍKLAD1).

FNC1 mód v druhé pozici - označuje jiný (vlastní) způsob kódování, předem schválený AIM International. Navíc obsahuje identifikátor aplikace.

Segment 1			.....	Segment n			Terminator
indikátor módu 1	počet znaků	data	.....	indikátor módu n	počet znaků	data	

Tabulka 2.15: Způsob skládání dat ve smíšeném módu QR-kódu.

PŘÍKLAD1

Data:
`01049123456123459`- Podle identifikátoru 01 se jedná o GS1 číslo produktu
`15970331` - identifikátor 15 značí datum spotřeby (YYMMDD)
`30128` - identifikátor 30 je množství
`10ABC123` - identifikátor 10 - číslo dávky, proměnné délky

Data pro zakódování:
`0104912345123459159703313012810ABC123`
% - tento znak uvádí alfanumerická data. Pokud je použit jako znak je třeba uvést %%.

Bitové sekvence v symbolu:
`0101` - indikátor módu - FNC1 v první pozici
`0001` - indikátor módu - numerický mód
`0000011101` - počet znaků, 29
`01049123451234591597033130128` - data
`0010` - indikátor módu - alfanumerický mód
`000001001` - počet znaků, 9
`10ABC123` data

Terminátor

Signalizuje konec dat. Je to řada nulových bitů, jejichž délka závisí na délce zprávy. Pokud se zpráva blíží ke konci kapacity matice, doplní se nuly pouze do konce nebo se terminátor vynechá.

Převod proudu bitů na kódová slova

Jedno kódové slovo má 8 bitů (kromě posledního znaku v Mikro QR-kódu M1 a M3, které má 4 bity).

Proud bitů se rozdělí do těchto kódových slov. Pokud data nekončí na hranici kódových slov, přidávají se vycpávkové bity o hodnotě 0 až do konce kódového slova.

Zbytek kapacity se doplní až do konce podle úrovně korekce chyby slovy 11101100 a 00010001 střídavě. U matic s posledním slovem délky 4 bity se doplní 0000.

Poté se doplní kódová slova pro korekci chyb.

2.4.5. Korekce chyby čtení QR-kódu

QR-kód využívá Reed-Solomonovo kódování pro detekci a korekci chyb. Je vygenerováno pár kódovacích slov, která jsou přidána do sekvence datových kódovacích slov.

PŘÍKLAD2
Verze 2-L má kapacitu 44 kódovacích slov; z čeho 34 jsou datové a 10 je korekčních.
$d = 10$
Z tabulky ve specifikaci [29] lze vyčíst, že kapacita korekce chyb jsou 4 chyby
$t = 4$
Počet výmazů je nula:
$e = 0$
Po doplnění rovnice:
$0 + 2 \times 4 = 10 - p$
Není zde $\leq$, jelikož se hledá mezní bod.
Pro vyvážení rovnice se musí $p = 2$.
Z toho plyne, že zbývající dvě kódová slova jsou schopny detekovat, ale ne opravit jakékoliv další chyby.
Po detekování páté chyby je dekódování označené za chybné.

Existují čtyři úrovně korekce chyby, které byly již zmíněny v tabulce 2.13.

Opravit se dají výmazy (chybné kódové slovo na známém místě) a chyby (chybné kódové slovo na neznámém místě). Výmazy jsou neskenované nebo nedekódovatelné symboly. Chyby jsou špatně dekódované symboly.

Obecně platí rovnice:

$$e + 2t \leq d - p \tag{2.2}$$

e - počet výmazů

t - počet chyb

d - počet korekčních kódových slov

p - počet kódových slov pro detekci chyby

Použití rovnice znázorňuje PŘÍKLAD2.

Generování chybových korekčních slov

Tato část nebude popsána s použitím specifikace, jelikož ta je na informace poněkud skoupá, ale s použitím zdroje, jenž zpracovává informace ze specifikace a poznatky z matematiky kolem generování korekčních slov [6].

Specifikace uvádí, že generování korekčních slov probíhá na základě bitové operace modulo 2_d a bitový modulo 100011101_b. To znamená, že se použije Galoisovo těleso (konečné těleso) 2^8.

Datová kódovací slova jsou koeficienty datového polynomu $d(x)$. Přičemž koeficient nejvyšší mocniny je první datové kódovací slovo a koeficient nejnižší mocniny je poslední datové kódovací slovo před prvním korekčním slovem.

Korekční slova samotná jsou zbytkem po dělení datového polynomu $d(x)$ polynomem $g(x)$, který se jmenuje generující polynom. Koeficient nejvyšší mocniny zbytku je první korekční slovo a koeficient nulové mocniny je poslední korekční slovo bloku.

PŘÍKLAD3
$2^8 = 256\hat{}285 = 29$
Další násobek dvěma bude:
$2^9 = 2^8 \cdot 2 = 29 \cdot 2 = 58$
Pokračuje:
$2^{10} = 2^9 \cdot 2 = 58 \cdot 2 = 116$
$2^{11} = 2^{10} \cdot 2 = 116 \cdot 2 = 232$
Až se vyskytne další číslo větší než 255:
$2^{12} = 2^{11} \cdot 2 = 232 \cdot 2 = 464\hat{}285 = 205$

Aritmetika Galoisova tělesa

QR-kód používá Galoisovo těleso 255. Značí se $GF(255)$ (*Galois Field*) nebo $GF(2^8)$.

Galoisovo těleso je množina konečného počtu prvků.

$GF(255)$ obsahuje pouze celá čísla 0-255, což znamená, že obsahuje čísla, která lze zobrazit osmibitovým číslem. 255 odpovídá binárnímu 11111111.

Aby byly výsledky operací s Galoisovými tělesy opět jejich částí, musí se operovat s operací modulo.

V aritmetice Galoisových těles jsou kladné i záporné hodnoty totéž:

$$-m = m \tag{2.3}$$

To znamená, že sčítání a odčítání je zaměnitelná operace a v případě použití bitové operace modulo 2 se jedná o funkci XOR.

$$(1+1) mod 2 = 2 \, mod 2 = 0 \tag{2.4}$$
$$1\hat{}1 = 0$$
$$(0+1) mod 2 = 2 \, mod 2 = 1 \tag{2.5}$$
$$0\hat{}1 = 1$$

Pro účely použití u QR-kódu je sčítání a odčítání považováno za operaci XOR.

Aritmetika bitového modulo 100011101_b

100011101_b je binární číslo odpovídající 285_d dekadických.

Použití u generování korekčních slov je, že když je číslo větší než 255_d, mělo by být XOR-ováno s 285_d (100011101_b). Vede na neintuitivní hodnoty do 255_d.

Použití vysvětluje PŘÍKLAD3.

Další možnou operací je modulo exponentu. Jelikož je GF(256) cyklická, může se zapsat toto:

$$2^{334} = 2^{334 \, mod \, 255} = 2^{79} \tag{2.6}$$

Této operace bude využito později.

PŘÍKLAD4
Uvažuje se g(x) 2. stupně.
$(x - \alpha^0) * (x - \alpha^1)$
po roznásobení:
$x^2 + \alpha^0 x + \alpha^1 x + \alpha^1$
sečtou se členy s x^1:
$x^2 + (\alpha^0 + \alpha^1)x + \alpha^1$
dosazení (sčítání u GF(256) je funkce XOR (^):
$x^2 + (1\text{^}2)x + 2$
$x^2 + 3x + 2$
převedení zpětně do notace α:
$\alpha^0 x^2 + \alpha^{25} x^1 + \alpha^1 x^0$ ... toto je generující polynom dvou korekčních slov.
($3 = \alpha^{25}$ vyplývá z aritmetiky GF(256))

Generující polynom

Tento polynom je vytvořen násobením: $(x - \alpha^0)...(x - \alpha^{n-1})$, kde n je počet korekčních slov a α je rovna 2.

Platí zde aritmetika modulo 2, aritmetika bytového modulo 100011101 a aritmetika Galoisova tělesa. Takže po vynásobení členů při sčítání koeficientů platí XOR. Objasnění postupu je provedeno pomocí PŘÍKLAD4.

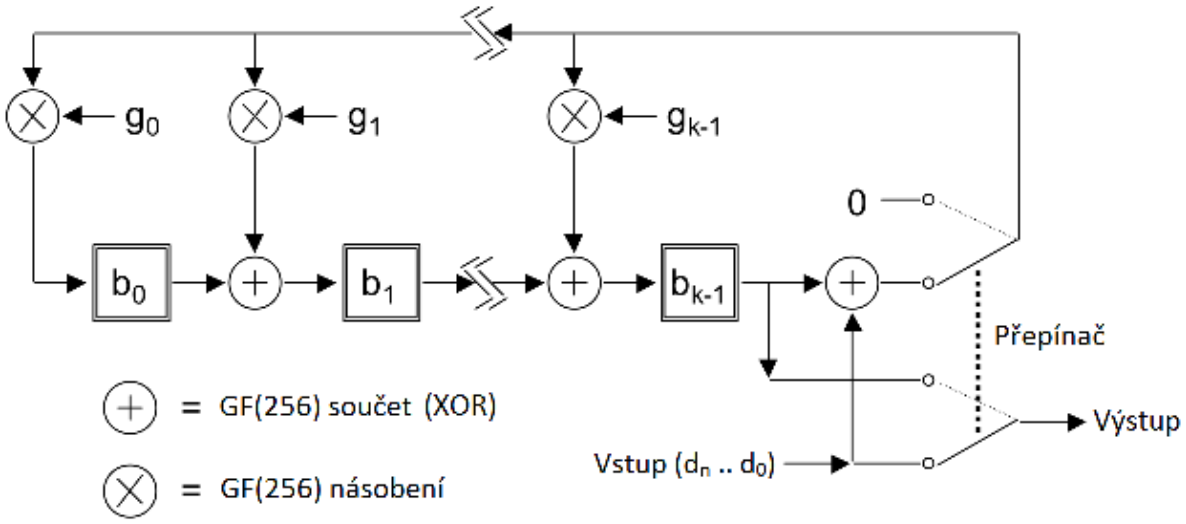
Všechny generátorové polynomy jsou dostupné ve specifikaci v příloze A dokumentace [29]. Nejsou tam dosazeny mocniny 2, ale ty se dají tabulkovat nebo počítat.

Generování korekčních slov

Je potřebný datový polynom $d(x)$ a generující polynom g(x). Dále probíhá dělení polynomů $d(x)$ polynomem $g(x)$.

Proces dělení

1. Vynásobení $d(x)$ x^n, kde n je stupeň $g(x)$. Je to proto, aby polynom nebyl příliš nízkého stupně.
2. Vynásobení $g(x)$ x^m, kde m je stupeň $d(x)$.
3. Dělení má tolik kroků, kolik členů má $d(x)$. Stupeň zbytku po dělení je o jeden stupeň nižší než stupeň $g(x)$ a jeho koeficienty jsou korekční slova.
4. Násobí se $g(x)$ nejvyšším koeficientem $d(x)$.
5. Sečte se výsledný polynom z 4. kroku s $d(x)$. Sečtením je myšlená operace XOR.
6. $g(x)$ se vydělí x, aby měl stejný stupeň jako polynom z předcházejícího kroku.
7. Vynásobí se $g(x)$ nejvyšším koeficientem výsledného polynomu z kroku 5. Pokud se vyskytne číslo větší než 255, aplikuje se modulo. Lze použít i notaci s α pro lepší přehlednost.



Obrázek 2.9: Implementace dělení polynomů jako dělicí cyklus. [29]

8. Sečte se výsledný polynom ze 7. kroku s $d(x)$. Sečtením je myšlená operace XOR.
9. Kroky 6, 7 a 8 se opakují, dokud polynom z 8. kroku není nižšího stupně než generující polynom.
10. Koeficienty zbylého polynomu jsou korekční slova.

Dělící cyklus

Algoritmus dělení polynomů uvedený výše se dá implementovat jako dělící cyklus zobrazený na obrázku 2.9. Tento způsob doporučuje specifikace [29].

Místo koeficientů $g_0..g_{k-1}$ se doplní koeficienty generujícího polynomu $g(x)$, s tím, že g_0 je koeficient nulové mocniny a k je stupeň polynomu $g(x)$.

Proměnné $b_0..b_{k-n}$ jsou registry.

Na počátku se přepínač drží v dolní pozici a okruh je připojen na vstup. Na vstup jsou poslány koeficienty datového polynomu $d(x)$. Pořadí koeficientů je $d_n..d_0$, kde n je stupeň polynomu $d(x)$.

Registry b se inicializují na nulu.

V první řadě se provede prakticky krok 4 z předešlého postupu a jakoby se polynom $g(x)$ vynásobí nejvyšším koeficientem $d(x)$.

Byl vynechán koeficient g_k, protože stejně se po operaci XOR vynuluje.

Nyní jsou koeficienty vynásobeného $g(x)$ v registrech b. Nejvýznamnější koeficient polynomu z kroku 4 procesu je nyní v registru b_{k-1}.

Nyní se polynom $g(x)$ násobí sečtenou (XOR) hodnotou koeficientu d_{n-1} a hodnotou registru b_{k-1}, což odpovídá provedení sečtení (XOR) hodnot z 5. bodu.

Nyní jsou v okruhu hodnoty ze 7. bodu postupu dělení. Ty se přičtou (XOR) k hodnotám v posunutém registru.

Postupně se vše sečte (XOR) jak má, ale ne v jednom kroku jako v předešlém příkladě, ale postupně.

Vše se opakuje, až projde celý polynom $d(x)$.

Na konci se vyčtou z registrů korekční slova.

PŘÍKLAD5
Algoritmus dělení polynomů.
Uvažují se polynomy:
$g(x) = g_2x^2 + g_1x + g_0$
$d(x) = d_2x^2 + d_1x + d_0$
Rozšíří se stupeň:
$g'(x) = g_2x^4 + g_1x^3 + g_0x^2$
$d'(x) = d_2x^4 + d_1x^3 + d_0x^2$
Polynom $g'(x)$ se vynásobí d_2:
$g_2 = 1$
$g''(x) = d_2x^4 + d_2g_1x^3 + d_2g_0x^2$
Sčítají se polynomy $g''(x)$ a $d'(x)$:
$g'''(x) = (d_2 \oplus d_2)x^4 + (d_2g_1 \oplus d_1)x^3 + (d_2g_0 \oplus d_0)x^2$
$g'''(x) = (d_2g_1 \oplus d_1)x^3 + (d_2g_0 \oplus d_0)x^2$

Násobení $g'(x)$ koeficientem $(d_2g_1 \oplus d_1)$ a snížení řádu:
$g^{(4)} = (d_2g_1 \oplus d_1)x^3 + (d_2g_1 \oplus d_1)g_1x^2 + (d_2g_1 \oplus d_1)g_0x^1$
Sčítají se polynomy $g^{(4)}$ a $g'''(x)$:
$g^{(5)} = ((d_2g_1 \oplus d_1) \oplus (d_2g_1 \oplus d_1))x^3 + ((d_2g_1 \oplus d_1)g_1 \oplus (d_2g_0 \oplus d_0))x^2 + (d_2g_1 \oplus d_1)g_0x^1$
$g^{(5)} = ((d_2g_1 \oplus d_1)g_1 \oplus (d_2g_0 \oplus d_0))x^2 + (d_2g_1 \oplus d_1)g_0x^1$

Násobení $g'(x)$ koeficientem $(((d_2g_1 \oplus d_1)g_1 \oplus (d_2g_0 \oplus d_0)))$ a snížení řádu:
$g^{(6)} = ((d_2g_1 \oplus d_1)g_1 \oplus (d_2g_0 \oplus d_0))x^2 + ((d_2g_1 \oplus d_1)g_1 \oplus (d_2g_0 \oplus d_0))g_1x + ((d_2g_1 \oplus d_1)g_1 \oplus (d_2g_0 \oplus d_0))g_0$
Sčítají se polynomy $g^{(5)}$ a $g^{(6)}$:
$g^{(7)} = (((d_2g_1 \oplus d_1)g_1 \oplus (d_2g_0 \oplus d_0)) \oplus ((d_2g_1 \oplus d_1)g_1 \oplus (d_2g_0 \oplus d_0)))x^2 +$
$+ (((d_2g_1 \oplus d_1)g_1 \oplus (d_2g_0 \oplus d_0))g_1 \oplus (d_2g_1 \oplus d_1)g_0x^1 + ((d_2g_1 \oplus d_1)g_1 \oplus (d_2g_0 \oplus d_0))g_0$
$g^{(7)} = (((d_2g_1 \oplus d_1)g_1 \oplus (d_2g_0 \oplus d_0))g_1 \oplus (d_2g_1 \oplus d_1)g_0x^1 + ((d_2g_1 \oplus d_1)g_1 \oplus (d_2g_0 \oplus d_0))g_0$

Zbytek po dělení je:
$(((d_2g_1 \oplus d_1)g_1 \oplus (d_2g_0 \oplus d_0))g_1 \oplus (d_2g_1 \oplus d_1)g_0x^1 + ((d_2g_1 \oplus d_1)g_1 \oplus (d_2g_0 \oplus d_0))g_0$

Nejlépe je vše vidět na PŘÍKLADU 5 a PŘÍKLADU 6.

PŘÍKLAD 5 vypadá dosti komplikovaně, ale nyní bude v PŘÍKLADU 6 předveden algoritmus dělícího cyklu a bude snaha dosáhnout stejného výsledku.

Jak je vidět z PŘÍKLADU 6, algoritmus dělícího cyklu dal stejný výsledek jako algoritmus dělení polynomu polynomem předvedený v PŘÍKLADU 5.

2.4.6. Finální sekvence kódovacích slov QR-kódu

Počet kódovacích slov by měl být vždy rovný kapacitě matice.

1. Rozdělení sekvence datových kódovacích slov na bloky. (Podle specifikace s uvážením typu a úrovně korekce chyb)

2. Pro každý blok spočítat sekvenci korekčních slov.

PŘÍKLAD6
Algoritmus dělícího cyklu. Uvažují se polynomy:
$g(x) = g_2x^2 + g_1x + g_0$
$d(x) = d_2x^2 + d_1x + d_0$
V cyklu se vyskytují pouze konstanty g_0 a g_1.
Inicializace registrů:
$b_0 = 0$
$b_1 = 0$

Do cyklu vstupuje $(0 \oplus d_2)$
Registry:
$b_0 = d_2g_0$
$b_1 = d_2g_1$

Do cyklu vstupuje $(d_2g_1 \oplus d_1)$
Registry:
$b_0 = (d_2g_1 \oplus d_1)g_0$
$b_1 = (d_2g_1 \oplus d_1)g_1 \oplus b_0(t-1)$
$b_1 = (d_2g_1 \oplus d_1)g_1 \oplus d_2g_0$

Do cyklu vstupuje $(((d_2g_1 \oplus d_1)g_1 \oplus d_2g_0) \oplus d_0)$
Registry:
$b_0 = (((d_2g_1 \oplus d_1)g_1 \oplus d_2g_0) \oplus d_0)g_0$
$b_1 = (((d_2g_1 \oplus d_1)g_1 \oplus d_2g_0) \oplus d_0)g_1 \oplus b_0(t-1)$
$b_1 = (((d_2g_1 \oplus d_1)g_1 \oplus d_2g_0) \oplus d_0)g_1 \oplus ((d_2g_1 \oplus d_1)g_0)$

Sestaví se polynom $b_1x + b0$:
$((((d_2g_1 \oplus d_1)g_1 \oplus d_2g_0) \oplus d_0)g_1 \oplus ((d_2g_1 \oplus d_1)g_0))x + ((((d_2g_1 \oplus d_1)g_1 \oplus d_2g_0) \oplus d_0)g_0)$
Po upravě a roznásobení:
$((d_2g_1^2 \oplus d_1g_1 \oplus d_2g_0 \oplus d_0)g_1 \oplus (d_2g_0g_1 \oplus d_1g_0))x + ((d_2g_1^2 \oplus d_1g_1 \oplus d_2g_0 \oplus d_0)g_0)$
$(d_2g_1^3 \oplus d_1g_1^2 \oplus d_2g_0g_1 \oplus d_0g_1 \oplus d_2g_0g_1 \oplus d_1g_0)x + (d_2g_0g_1^2 \oplus d_1g_0g_1 \oplus d_2g_0^2 \oplus d_0g_0)$

Zbytkový polynom z PŘÍKLADU 5:
$(((d_2g_1 \oplus d_1)g_1 \oplus (d_2g_0 \oplus d_0))g_1 \oplus (d_2g_1 \oplus d_1)g_0x + ((d_2g_1 \oplus d_1)g_1 \oplus (d_2g_0 \oplus d_0))g_0$
Po upravě a roznásobení:
$((d_2g_1^2 \oplus d_1g_1 \oplus d_2g_0 \oplus d_0)g_1 \oplus (d_2g_0g_1 \oplus d_1g_0)x + (d_2g_1^2 \oplus d_1g_1 \oplus d_2g_0 \oplus d_0)g_0$
$(d_2g_1^3 \oplus d_1g_1^2 \oplus d_2g_0g_1 \oplus d_0g_1 \oplus d_2g_0g_1 \oplus d_1g_0)x + (d_2g_0g_1^2 \oplus d_1g_0g_1 \oplus d_2g_0^2 \oplus d_0g_0)$

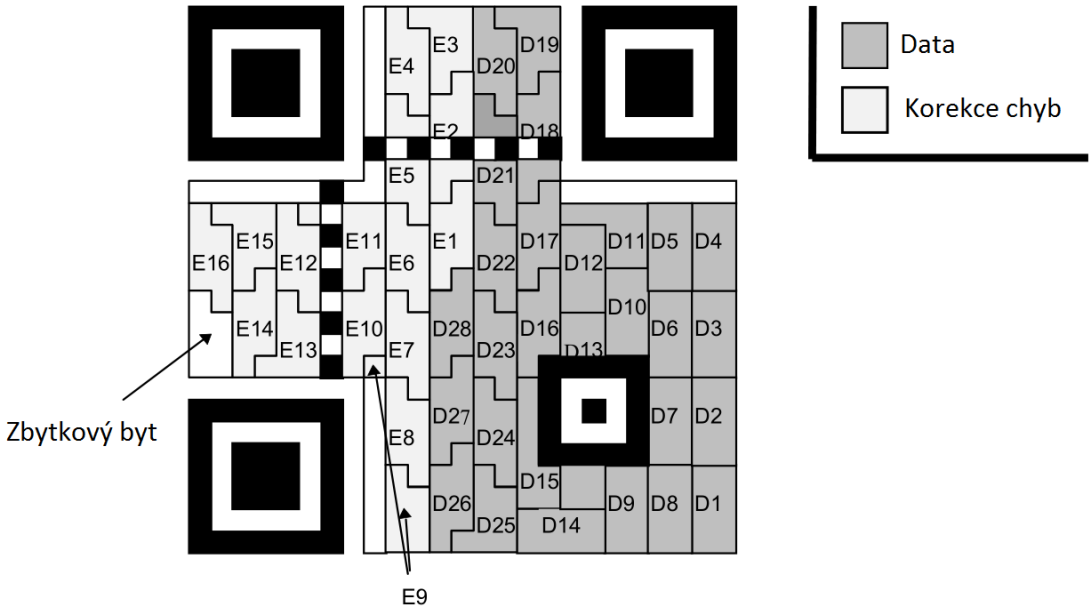
3. Sestrojení Zprávy s pořadím, které je znázorněno v tabulce 2.16. Zapisuje se nejdříve první sloupec, pak druhý, třetí atd.
   Podle tabulky 2.16 bude řetězec dat: D_1, D_{12}, D_{23}, D_{35}, D_2, D_{13}, D_{24}, D_{36}, ... D_{11}, D_{22}, D_{33}, D_{45}, D_{34}, D_{46}, E_1, E_{23}, E_{45}, E_{67}, E_2, E_{24}, E_{46}, E_{68}, ... E_{22}, E_{44}, E_{66}, E_{88}.

	Datová slova					Korekční slova			
Blok 1	D_1	D_2	.....	D_{11}		E_1	E_2	.....	E_{22}
Blok 2	D_{12}	D_{13}	.....	D_{22}		E_{23}	E_{24}	.....	E_{44}
Blok 3	D_{23}	D_{24}	.....	D_{33}	D_{34}	E_{45}	E_{46}	.....	E_{66}
Blok 4	D_{35}	D_{36}	.....	D_{45}	D_{46}	E_{67}	E_{68}	.....	E_{88}

Tabulka 2.16: Nastínění sestavení zprávy pro verzi 5-H QR-kódu.

2.4.7. Umístění kódovacích slov v matici QR-kódu

Slova jsou umístěna v matici jeden za druhým. Začíná se v pravém dolním rohu a pokračuje se nahoru a pak zase dolů. Nejlépe toto znázorňuje obrázek 2.10.



Obrázek 2.10: Ukázka pořadí slov v matici. [29]

Umístění bitů v matici se těžko popisuje slovně, nejlépe to představuje obrázek 2.11, kde je zobrazeno více kritických míst v matici a způsob jejich zaplnění bity.

Obecně platí, že zápis je prováděn zprava doleva a nahoru a dolů.

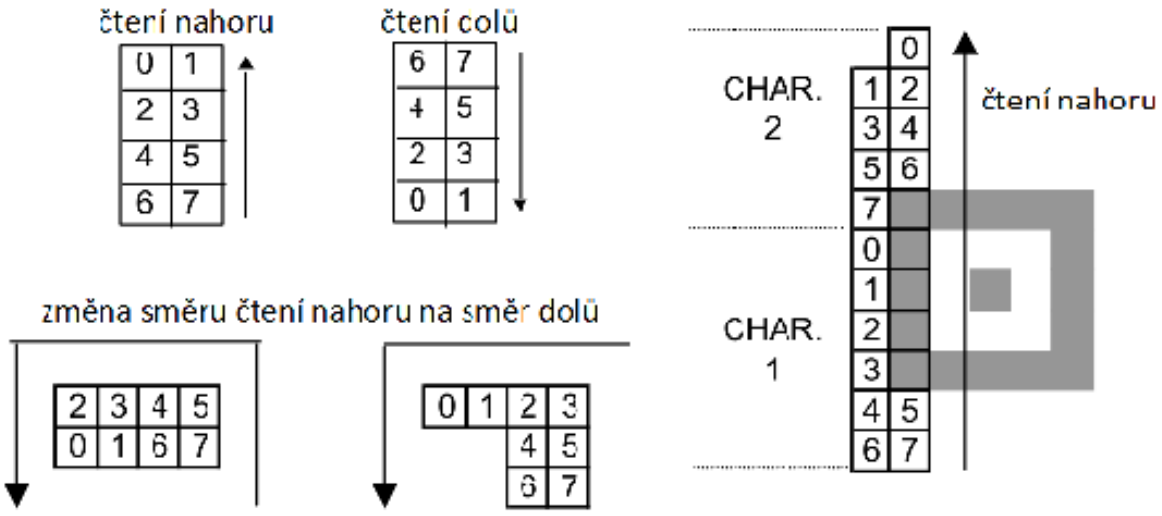
Základní pravilo zní, že nejvýznamnější bit slova (označený jako 7) má být umístěn v prvním volném umístění bitu, které logicky následuje za nejméně významným bitem (označeným jako 0) předchozího slova.

Nejjednodušší je představit si vše jako proud bitů a zapisovat bit po bitu podle pravidla zprava doleva a nahoru a dolů.

2.4.8. Maskování dat v matici QR-kódu

Aplikuje se, aby byla tmavá a světlá místa zastoupená stejně.

1. Neaplikuje se na funkční vzory.
2. Aplikuje se operace XOR, která barevně reverzuje oblast matice.
3. Vyhodnotí se nežádoucí jevy na každé transformaci.



Obrázek 2.11: Příklad umístění bitů slov ve vzorech v matici. [29]

4. Vybere se maska s nejmenším počtem nežádoucích jevů.

Specifikace uvádí osm maskovacích vzorů. Masky zobrazuje obrázek 2.12. Pro Mikro QR-kód jsou masky jen čtyři. Jejich celkový přehled je dostupný ve specifikaci.

Vyhodnocení masky

Po aplikaci všech masek se počítá skóre. Čím vyšší je skóre, tím méně akceptovatelný je výsledek.

1. Počet modulů stejné barvy za sebou v jedno řádku nebo sloupci. Pokud je takových modulů $5+i$, tak se skóre navýší o $3+i$, přičemž i znamená, o kolik delší je taková oblast než 5.

2. Blok modulů stejné barvy velikosti $m \times n$. Skóre se navýší o $3 \times (m-1) \times (n-1)$.

3. Přítomnost vzoru 1:1:3:1:1 (tmavý:světlý:tmavý:světlý:tmavý) v řádku nebo sloupci, kterou předchází nebo následuje oblast čtyř světlých modulů. Skóre se navýší o 40.

4. Proporce tmavých modulů ke světlým modulům. $50 \pm (5 \times k)\%$, kde k je velikost odchylky od 50% v krocích 5%. Skóre se navýší o $10 \times k$.

2.4.9. Informace o formátu QR-kódu

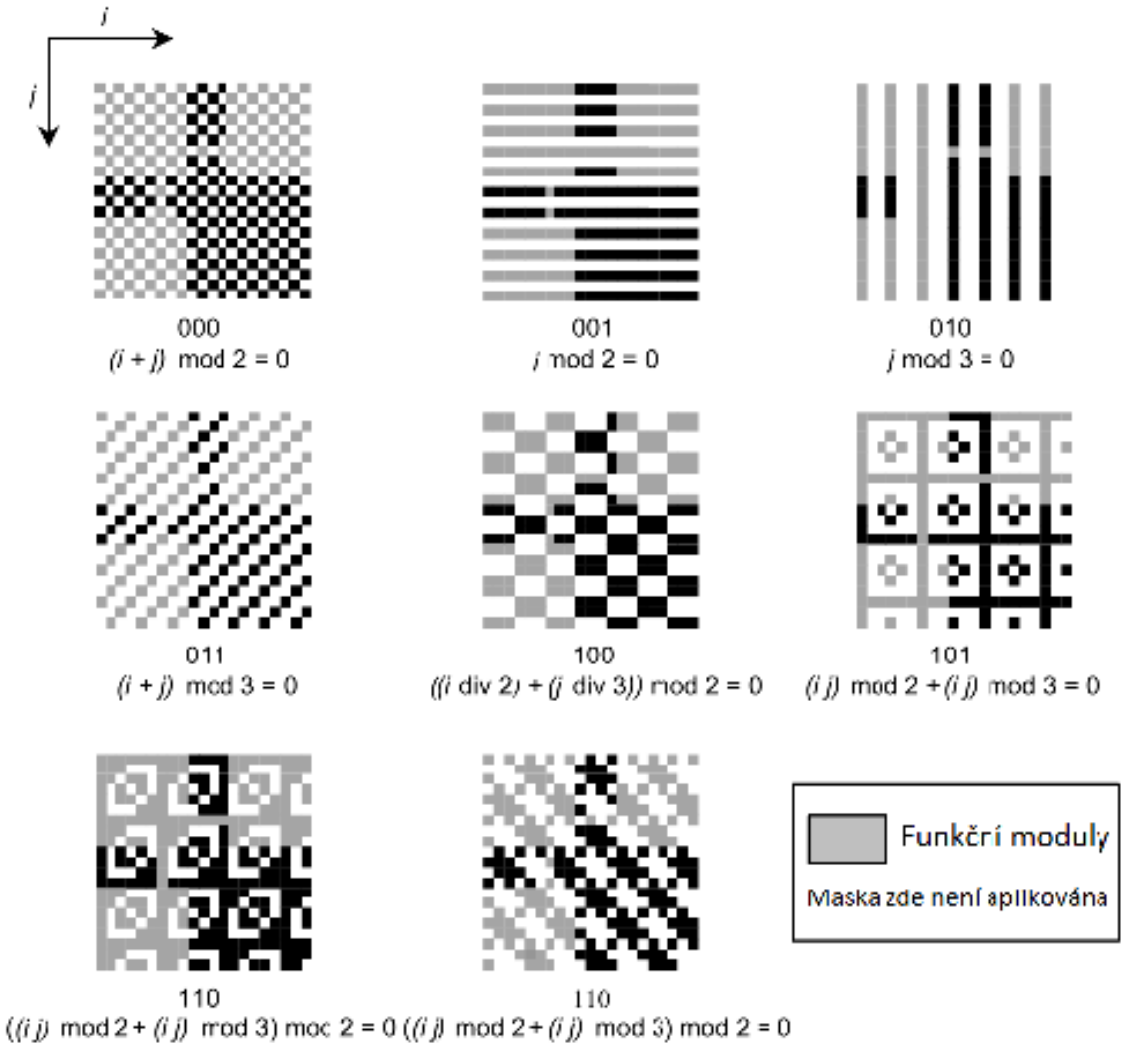
Sekvence o délce 15 bitů. 5 datových a 10 korekčních počítaných (15, 5) BCH kódem.

První dva datové bity určují úroveň korekce chyb. L - 01; M - 00; Q - 11; H - 10.

Další tři bity určují masku. Označení je vidět na obrázku 2.12.

Umístění informací o formátu ukazuje obrázek 2.13. Jelikož tyto informace jsou nezbytné pro dekódování, jsou v matici umístěny dvakrát. 14 je nejvýznamnější bit a 0 je nejméně významný bit.

Pro Mikro QR-kód jsou první tři bity data o verzi a úrovni korekce chyb a další dva jsou pro označení jedné ze čtyř masek. Korekční bity se vypočtou stejně, ale použije se jiná posloupnost pro XOR. Data jsou do matice umístěna jen jednou.



Obrázek 2.12: Masky pro maskování matice. [29]

Výpočet korekce chyby BCH kódem

BCH - Bose-Chaudhuri-Hocquenghem (15,5) kód je založen na polynomu:

$$G(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1 \tag{2.7}$$

Datové bity se převedou na polynom $D(x)$ s nejnižší mocninou $(15 - 5)$. Dělí se polynomy: $D(x)/G(x)$.

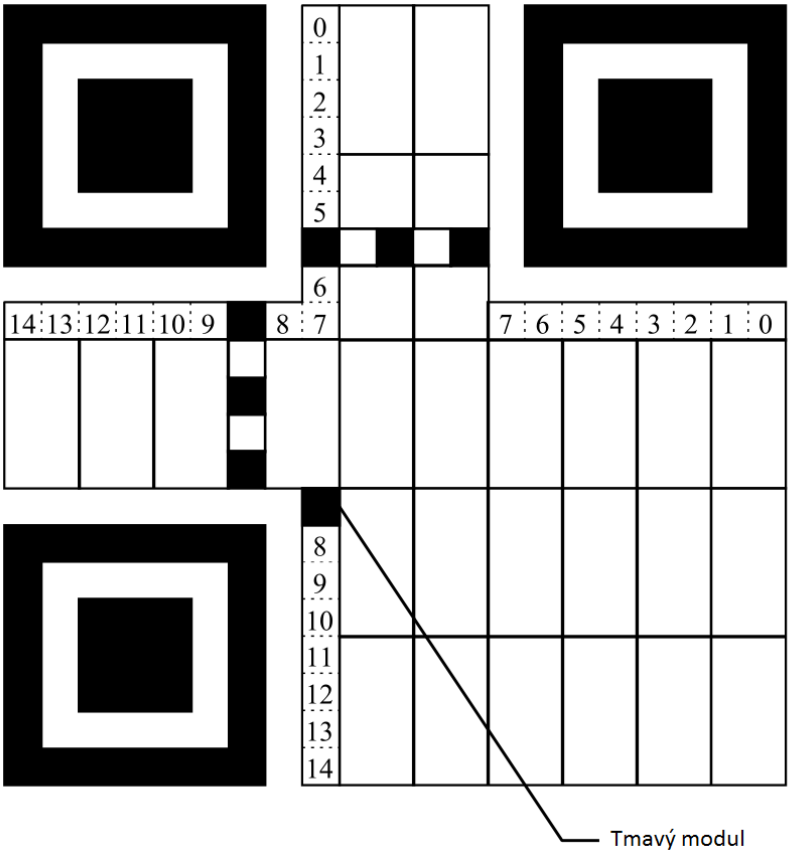
Zbytek po dělení se spojí s datovými bity a celek se XOR-uje s 10101 00000 10010 nebo pro Mikro QR-kód s 10001 00010 00101.

Postup bude znázorněn PŘÍKLADEM 7, na kterém bude vše vysvětleno.

2.4.10. Informace o verzi QR-kódu

Informace o verzi jsou ve verzi 7 a vyšší. Tvoří je sekvence 18 bitů - 6 datových a 12 korekčních počítaných (18, 6) Golayovým kódem. Výpočet se nemusí provádět, hodnoty se dají tabulkovat.

Informace jsou opět umístěny v matici dvakrát. Vše je znázorněno na obrázku 2.14. Nejvýznamnější bit je 14, nejméně významný bit je 0.



Obrázek 2.13: Umístění 15 bitů s informacemi o formátu. [29]

PŘÍKLAD7

Úroveň korekce chyby M; maska `101`

Binární data: `00101`
Polynom: $D(x) = x^2 + 1$
Zvednutí mocniny $D(x) = x^{12} + x^{10}$
Dělení: $\frac{D(x)}{G(x)} = (x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1)x^2 + (x^7 + x^6 + x^4 + x^3 + x^2)$
Toto se dá zapsat také jako bitová operace:

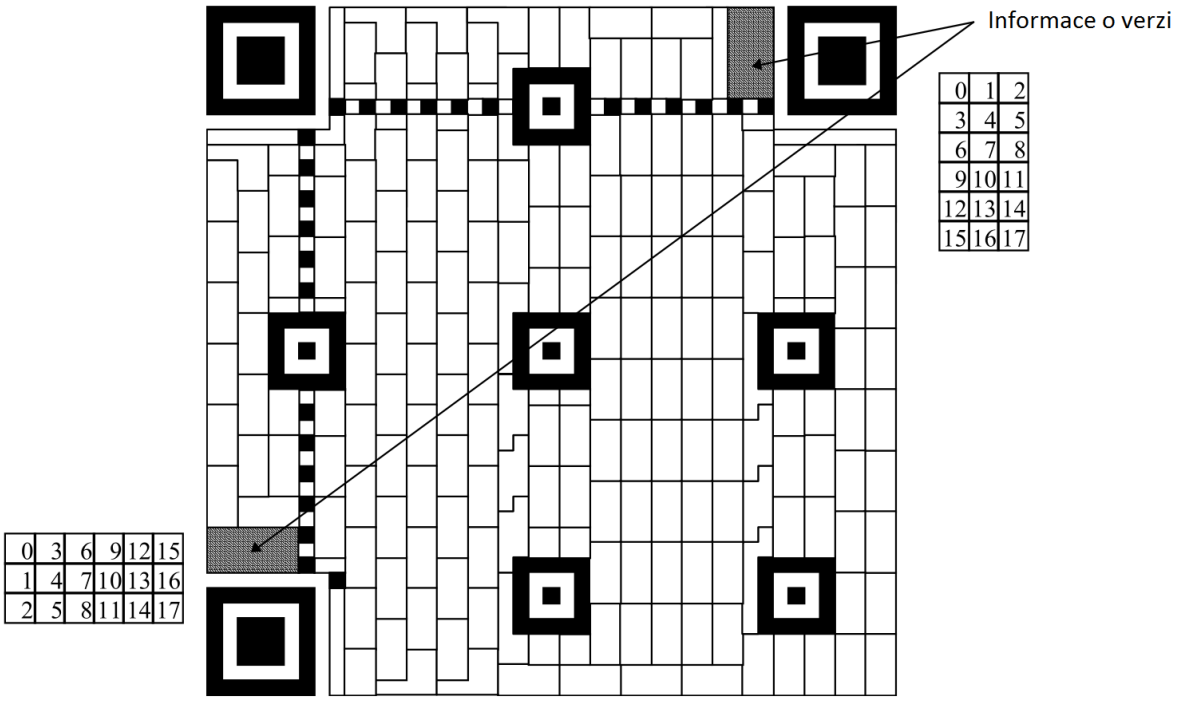
```
001010000000000
  1010011011100
      0011011100
```

Zbytek: $(x^7 + x^6 + x^4 + x^3 + x^2) \rightarrow$ `00110 11100`
Spojení dat a korekce: `00101 + 00110 11100` → `00101 00110 11100`
XOR s maskou: `10101 00000 10010`
Výsledek: `10000 00110 01110`

Výpočet korekce pomocí Golayova kódu

Golayův kód je založen na polynomu:

$$G(x) = x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^5 + x^2 + 1 \tag{2.8}$$



Obrázek 2.14: Umístění 18 bitů s informacemi o verzi. [29]

PŘÍKLAD8

Verze 7

Binární data: `000111`
Polynom: $D(x) = x^2 + x + 1$
Zvednutí mocniny: $D(x) = x^{14} + x^{13} + x_{12}$
Dělení: $\frac{D(x)}{G(x)} = (x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^5 + x^2 + 1)x^2 + (x^{11} + x^{10} + x^7 + x^4 + x^2)$
Zbytek: $(x^{11} + x^{10} + x^7 + x^4 + x^2) \rightarrow$ `110010 010100`
Spojení dat a korekce: `000111 + 110010 010100` $\rightarrow$ `000111 110010 010100`

Datové bity se převedou na polynom $D(x)$ s nejnižší mocninou $(18 - 6)$. Dělí se polynomy: $D(x)/G(x)$.

Postup je znázorněn PŘÍKLADEM 8, na kterém je vše vysvětleno.

3. Oprava a detekce chyb při čtení QR kódu

V předchozí kapitole byl popsán postup zakódování dat do matice QR-kódu. Nyní bude věnován prostor způsobu opravy chyb při dekódování matice QR-kódu.

3.1. Detekce a oprava chyb v informacích o formátu

Zde je věc jednoduchá. Nejprve se provede operace XOR s odpovídající maskou (masky jsou uvedeny u popisu kódování). Nyní jsou k dispozici data spojená se zbytkem po dělení dat a generujícího polynomu. Toto lze označit jako polynom $R(x)$.

Dále je známo, že existuje 32 formátů QR-kódu. To znamená, že existuje 32 správných $R'(x)$ polynomů. Tyto polynomy mají Hammingovu vzdálenost rovnou 7 (liší se v sedmi bitech). To dává možnost opravy 3 bitů.

Jako správný se bere polynom $R'(x)$, který má nejmenší Hammingovu vzdálenost k přečtenému polynomu $R(x)$

3.2. Detekce a oprava chyb v informacích o verzi

Z kódu je přečten polynom $R(x)$.

Existuje omezený počet 34 správných polynomů $R'(x)$, které mají Hammingovu vzdálenost rovnou 8.

Opět jako u informace o formátu se jako správný bere polynom $R'(x)$, který má nejmenší Hammingovu vzdálenost k přečtenému polynomu $R(x)$.

3.3. Detekce a oprava chyb v datech

Nejprve je třeba uvést, že informace v této kapitole vycházejí z dokumentace ke QR-kódu [29], článku *Reed–Solomon codes for coders — Wikiversity* [23] a článku *Reed–Solomon error correction — Wikipedia, The Free Encyclopedia* [22]

Zpráva se považuje za polynom. Pokud zpráva obsahuje N slov, pak polynom bude $R(x) = r_0 + r_1x + r_2x^2 + ... + r_{N-1}x^{N-1}$. Je třeba pamatovat, že koeficienty r_0 - r_{N-1} jsou elementy $GF(256)$.

Dekódování zprávy s Reed-Solomnovou opravou chyb sestává z více kroků. V první řadě je potřeba vypočítat n *syndromů* zprávy. Přičemž n je počet kódových slov volných pro korekci chyby.

Pokud by se brala v úvahu verze 1-M, kde počet kódových slov $c = 26$, počet datových kódových slov $k = 16$ a kapacita korekce chyb $t = 4$, potom $n = c - k - p$, kde p je počet kódových slov pro detekci chyby. Vše vychází z rovnice 2.2. V tomto případě $n = 26 - 16 - 2 = 8$.

Syndrom se vypočte z polynomu $R(x)$ podle předpisu 3.1.

$$S_k = R(\alpha^k) = r_0 + r_1\alpha^k + r_2\alpha^2 k + ... + r_{c-1}\alpha^{(c-1)k}, \tag{3.1}$$

kde k je číslo syndromu z rozsahu 1 až n a α je primitivní element GF(256), takže $\alpha = 2^8$.

Rozepsání pro příklad 1-M (3.2):

$$\begin{aligned} S_1 &= R(1) = r_0 + r_1 + r_2 + ... + r_{25} \\ S_2 &= R(\alpha) = r_0 + r_1\alpha + r_2\alpha^2 + ... + r_{25}\alpha^{25} \\ &\vdots \\ S_8 &= R(\alpha^7) = r_0 + r_7\alpha + r_2\alpha^1 4 + ... + r_{25}\alpha^{175}. \end{aligned} \tag{3.2}$$

Syndromy mají dvě zásadní vlastnosti: jsou dělitelné generujícím polynomem a jsou rovny nule.

Tedy pokud jsou syndromy rovny 0, je zpráva bez chyby. Pokud nulové nejsou, došlo ke změně. Jinak řečeno, syndromy $S(x) = s(x) + e(x)$, kde $s(x)$ je bezchybný syndrom a $e(x)$ je přičtený polynom, který zavádí chybu. Zde opět pozor, operátor + v syndromech je operace v Galiusově tělese, proto vychází 0.

Koeficienty polynomu $e(x)$ budou nulové pro mocniny, ve kterých není chyba, tudíž pro slova bez chyby.

Jestliže je známo, že bezchybný polynom $s(x) = 0$, potom $S(x) = 0 + e(x)$ a polynom $e(x)$ lze definovat jako:

$$e(x) = \sum_{i=1}^{\nu} e_{j_i} x^{j_i} = \sum_{i=1}^{\nu} e_{j_i} (\alpha^k)^{j_i}. \tag{3.3}$$

Cílem je nalézt počet chyb (ν), pozici chyb (j_i) a velikost chyb na těchto pozicích (e_{j_i}).

Nejprve je potřeba definovat lokátor chyby $X_i = \alpha^{j_i}$ a velikost chyby $Y_i = e_{j_i}$.

Potom lze syndrom napsat jako:

$$S_k = \sum_{i=1}^{\nu} Y_i X_i^k. \tag{3.4}$$

Toto dává soustavu rovnic danou maticově:

$$\begin{bmatrix} X_1^1 & X_2^1 & \cdots & X_\nu^1 \\ X_1^2 & X_2^2 & \cdots & X_\nu^2 \\ \vdots & \vdots & & \vdots \\ X_1^n & X_2^n & \cdots & X_\nu^n \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_\nu \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_n \end{bmatrix}. \tag{3.5}$$

Z této soustavy nelze vypočíst velikost chyby, pokud není známá její lokace. Pro tyto účely je zapotřebí sestavit polynom lokace chyby $\sigma(x)$:

$$\sigma(x) = \prod_{i=1}^{\nu} 1 - xX_i = 1 + \sum_{i=1}^{\nu} \sigma_i x^i. \tag{3.6}$$

Nuly tohoto polynomu $\sigma_1, \sigma_2, \ldots, \sigma_\nu$ jsou reciproké k poloze chyby $X_i = \sigma_i^{-1} = \alpha^{j_i}$.

Pro připomenutí nebo ujasnění, i je číslo chyby a j je číslo chybného slova. Potom j_i je označení slova, na kterém je i-tá chyba.

Dosadí-li se do polynomu $\sigma(x)$ jeho nula, potom

$$\sigma(\sigma_i) = \sigma(X_i^{-1}) = 1 + \sigma_1 X_i^{-1} + \sigma_2 X_i^{-2} + \cdots + \sigma_\nu X_i^{-\nu} = 0. \tag{3.7}$$

Pokud se nyní obě strany vynásobí $Y_i X_i^{k+\nu}$, výsledek bude stále nula:

$$Y_i X_i^{k+\nu}\sigma(X_i^{-1}) = Y_i X_i^{k+\nu} + \sigma_1 Y_i X_i^{k+\nu} X_i^{-1} + \sigma_2 Y_i X_i^{k+\nu} X_i^{-2} + \cdots + \sigma_\nu Y_i X_i^{k+\nu} X_i^{-\nu} = 0. \quad (3.8)$$

Pro připomenutí je třeba zmínit, že k je číslo syndromu a je z rozsahu 1 až n.
Po úpravách:

$$Y_i X_i^{k+\nu}\sigma(X_i^{-1}) = Y_i X_i^{k+\nu} + \sigma_1 Y_i X_i^{k+\nu-1} + \sigma_2 Y_i X_i^{k+\nu-2} + \cdots + \sigma_\nu Y_i X_i^{k} = 0. \quad (3.9)$$

Pokud se zesumují všechny tyto polynomy od $i = 1$ po $i = \nu$

$$\sum_{i=1}^{\nu} (Y_i X_i^{k+\nu} + \sigma_1 Y_i X_i^{k+\nu} X_i^{-1} + \sigma_2 Y_i X_i^{k+\nu} X_i^{-2} + \cdots + \sigma_\nu Y_i X_i^{k+\nu} X_i^{-\nu}) = 0, \quad (3.10)$$

pak lze tuto rovnici přepsat jako:

$$\sum_{i=1}^{\nu} (Y_i X_i^{k+\nu}) + \sigma_1 \sum_{i=1}^{\nu} (Y_i X_i^{k+\nu-1}) + \sigma_2 \sum_{i=1}^{\nu} (Y_i X_i^{k+\nu-2}) + \cdots + \sigma_\nu \sum_{i=1}^{\nu} (Y_i X_i^{k}) = 0. \quad (3.11)$$

Po zredukování lze napsat:

$$S_{k+\nu} + \sigma_1 S_{k+\nu-1} + \sigma_2 S_{k+\nu-2} + \cdots + \sigma_\nu S_k = 0 \quad (3.12)$$

a po změně pořadí a převedení členu na druhou stranu:

$$S_k \sigma_\nu + S_{k+1}\sigma_{\nu-1} + \cdots + S_{k+\nu-1}\sigma_1 = -S_{k+\nu}. \quad (3.13)$$

Celek lze maticově přepsat:

$$\begin{bmatrix} S_1 & S_2 & \cdots & S_\nu \\ S_2^2 & S_3 & \cdots & S_{\nu+1} \\ \vdots & \vdots & & \vdots \\ S_\nu & S_{\nu+1} & \cdots & S_{2\nu-1} \end{bmatrix} \begin{bmatrix} \sigma_\nu \\ \sigma_{\nu-1} \\ \vdots \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} -S_{\nu+1} \\ -S_{\nu+2} \\ \vdots \\ -S_{\nu+\nu} \end{bmatrix}. \quad (3.14)$$

Zde vyvstává problém, že počet chyb nebyl ještě určen a určit dopředu ani nejde. Proto se nejprve bere jejich nejvyšší počet, a pokud je soustava řešitelná (determinant matice je nenulový), je počet chyb správný, pokud je soustava neřešitelná, zmenší se počet chyb o jedna a zkouší se znovu.

V pokračování předešlého příkladu, kde jsou předpokládány čtyři chyby, bude polynom $\sigma(x)$ vypadat takto:

$$\sigma(x) = \sigma_4 + \sigma_3 x + \sigma_2 x^2 + \sigma_1 x^3 + x^4 \quad (3.15)$$

a soustava rovnic bude následovna:

$$\begin{aligned} S_1\sigma_4 - S_2\sigma_3 + S_3\sigma_2 - S_4\sigma_1 + S_5 &= 0 \\ S_2\sigma_4 - S_3\sigma_3 + S_4\sigma_2 - S_5\sigma_1 + S_6 &= 0 \\ S_3\sigma_4 - S_4\sigma_3 + S_5\sigma_2 - S_6\sigma_1 + S_7 &= 0 \\ S_4\sigma_4 - S_5\sigma_3 + S_6\sigma_2 - S_7\sigma_1 + S_8 &= 0. \end{aligned} \quad (3.16)$$

Je třeba pamatovat, že při dosazení do polynomu $\sigma(x)$ je potřeba převést koeficienty na elementy $GF(256)$.

Získání lokátorů chyb z polynomu lokace chyb $\sigma(x)$

Jelikož víme, že chyby se nacházejí v poloze určené lokátorem reciprokým k nulám polynomu $\sigma(x)$, je potřeba vyřešit rovnici:

$$\sigma(\alpha) = 0. \tag{3.17}$$

Po vyřešení rovnice je známo $\alpha^{j_1} = X_1, \alpha^{j_2} = X_2, \ldots, \alpha^{j_\nu = X_\nu}$. Nyní je potřeba vyřešit rovnice obecně dané tvarem:

$$\alpha^{j_i} = X_i, \tag{3.18}$$

kde j je neznámá. většinou se to provádí tabulkovanými hodnotami, jelikož se nadále pracuje v prostoru GF(256) a hodnot je omezený počet.

Výpočet velikosti chyby

Velikost chyb je možné vypočítat z dosazení do soustavy rovnic 3.5. Nyní jsou známy jak syndromy $S_0, S_1, \ldots, S_{n-1}$, tak lokátory chyb X_i^k:

$$\begin{aligned} Y_1X_1 + Y_2X_2 + ... + Y_kX_\nu &= S_1 \\ Y_1X_1^2 + Y_2X_2^2 + ... + Y_kX_\nu^2 &= S_2 \\ &\vdots \\ Y_1X_\nu^\nu + Y_2X_\nu^\nu + ... + Y_kX_\nu^\nu &= S_\nu. \end{aligned} \tag{3.19}$$

Zde bude stačit tolik rovnic, kolik je chyb, takže i neznámých v rovnicích.

Pro vyřešení rovnic už jenom zbývá sestrojit polynom $e(x)$ na základě známosti j_i a e_i a odečíst ho od polynomu $R(x)$ pro získání původního polynomu $s(x)$.

Alternativní výpočet velikosti chyb pomocí Forneyho algoritmu

Informace jsou čerpány z článku *Forney algorithm From Wikipedia, the free encyclopedia* [20].

Forneyho algoritmus je založen na Lagrangeově interpolaci a postup výpočtu bude dále nastíněn.

Nejprve je potřeba spočítat polynom velikosti chyb:

$$\Omega(x) = S(x)\sigma(x)(\text{mod } x^n), \tag{3.20}$$

kde S(x) je parciální syndromický polynom:

$$S(x) = S_1x^1 + S_2x^2 + \cdots + s_nx^n. \tag{3.21}$$

Poté lze vypočítat velikosti chyb:

$$e_i = -\frac{X_i^{1-k}\Omega(X_i^{-1})}{\sigma'(X_i^{-1})}, \tag{3.22}$$

kde $\sigma'(x)$ je formální derivací polynomu lokace chyb $\sigma(x)$:

$$\sigma'(x) = \sum_{\nu}^{i=1} \sigma_i x^{i-1}. \tag{3.23}$$

Pokud se zvolí $k = 1$, lze psát:

$$e_i = -\frac{\Omega(X_i^{-1})}{\sigma'(X_i^{-1})}. \tag{3.24}$$

Řešení situace, pokud jsou přítomny výmazy (chyby se známou polohou)

Pokud je známá poloha chyby, zavede se polynom:

$$\gamma(x) = \prod (1 - x\alpha^{i_j}), \tag{3.25}$$

kde je lokace výmazu dána hodnotou i_j. Zde je polynom lokace chyb jíž hotov a nemusí se počítat. Proces výpočtu velikosti chyb se provede s polynomem $\gamma(x)$ místo polynomu $\sigma(x)$. Pokud jsou přítomny jak výmazy, tak chyby s neznámou lokací, definuje se polynom:

$$\psi(x) = \sigma(x)\gamma(x), \tag{3.26}$$

který je dále používán ve výpočtech místo polynomu $\sigma(x)$.

4. Současné metody nalezení kódu ve snímku

Obecně se nějaká oblast (objekt) hledá na základě porovnání se vzorem nebo na základě nějakých předem definovaných vlastností plochy (textury).

Porovnání se vzorem se často děje pomocí porovnání předem definovaných vlastností vzoru a části snímku. Matice se posouvá přes snímek a postupně se vše porovnává. Jelikož je třeba projít více velikostí posuvného okna, všechny možné jeho pozice a často i rotace, je tato metoda dosti výpočetně náročná a zdlouhavá.

U černobílých neboli také binárních vzorů lze sledovat hlavně hrany (ostré změny jasu), průměrný jas, dominantní úhel/úhly natočení hran.

Před počátkem hledání je vhodné obraz vyprahovat. Jednodušší je použití jednoho prahu pro celý obraz, ale v tom případě je nutno snímat v kontrolovaném a dobře a rovnoměrně nasvětleném prostředí tak, aby nedocházelo k velkým změnám kontrastu v obraze.

Při použití snímků z nekontrolovaného prostředí je zapotřebí použít dynamickou změnu prahu.

Všechny potřebné nástroje a metody budou probrány v následujících podkapitolách.

Potom budou uvedeny některé používané metody a algoritmy čtení kódů.

Jelikož se nepodařilo najít články týkající se IMB kódu, nebudou zde popsány žádné současné metody, ale pouze třídicí stroj, který tento kód používá pro třídění psaní.

4.1. Nástroje pro zpracování snímků

V této části budou vysvětleny základní nástroje, které se používají při hledání a čtení kódu ve snímku.

Základní věci jako histogram, barevné modely a jiné, nebudou vysvětlovány, předpokládá se minimální znalost problematiky.

4.1.1. Prahování snímků

Prahování spočívá v převedení obrazu s více úrovněmi jasu na obraz s pouze dvěma úrovněmi jasu. Nejčastěji se pak používá binární obraz, kde je jas pixelů reprezentován čísly 0 a 1. Jedna znamená přítomnost jasu (bílá barva) a nula značí nepřítomnost jasu (černá barva).

Obecně je funkce pro jas pixelu následující:

$$f(x,y) = \begin{cases} 1 & g(x,y) \geq p \\ 0 & g(x,y) < p \end{cases} \tag{4.1}$$

kde $f(x,y)$ je vyprahovaný obraz, $g(x,y)$ je původní obraz, p je velikost prahu a x a y jsou souřadnice v obraze.

Práh se může měnit v obraze dynamicky, v tom případě je i práh funkcí s hodnotou závislou na souřadnicích (obecně se používá termín lokální práh):

$$f(x,y) = \begin{cases} 1 & g(x,y) \geq p(x,y) \\ 0 & g(x,y) < p(x,y) \end{cases} \tag{4.2}$$

4.1.2. Určení velikosti prahu

Pokud se velikost prahu neurčí předem nebo se nehledá ručně metodou pokus omyl, je ji třeba dopočítat.

Nejjednodušší metoda je metoda hledání prahu ze statistických vlastností obrazu. Přepokládá se, že v tomto momentě se pracuje s obrazem v úrovních šedi nebo s jinou kombinací složek obrazu, které tvoří dvourozměrnou matici $x \times y$.

O statistických vlastnostech nejlépe vypovídá histogram. Většinou se prahují obrazy, kde je výrazný odstup popředí od pozadí. Vznikne bimodální histogram s dvěma maximy v histogramu a jako práh lze použít minimum mezi nimi.

Další možností je nalezení průměru nebo mediánu a tuto hodnotu použít jako práh. Tato metoda se spíše uplatní u lokálního prahu, kdy se pro každý pixel spočítá vlastní práh z okolí.

Například výpočet prahu průměrováním z n okolí by vypadal takto:

$$p(x,y) = \frac{1}{n^2} \sum_{\substack{x'=x-n \\ y'=y-n}}^{\substack{x+1 \\ y+1}} f(x',y') \tag{4.3}$$

Vyskytne se zde problém s okraji obrázku, kde výpočet prahu zabíhá mimo souřadnice. V tomto případě se buďto okraje vynechají, nebo se práh vypočte z menšího okolí.

4.1.3. Detekce hran ve snímku

Správné nalezení hran je důležitou součástí hledání kódu v obraze. Existuje více zdokumentovaných hranových filtrů. Všechny jsou založeny na hledání jasové diference v obraze.

Diference je prováděná pomocí operátorů, které říkají, jak se mají sčítat pixely v okolí počítaného bodu. Operátor je ve tvaru matice, která se posouvá obrazem a provádí se operace násobení hodnot pixelů hodnotami odpovídajícími buňkám matice a následně se vše sečte.

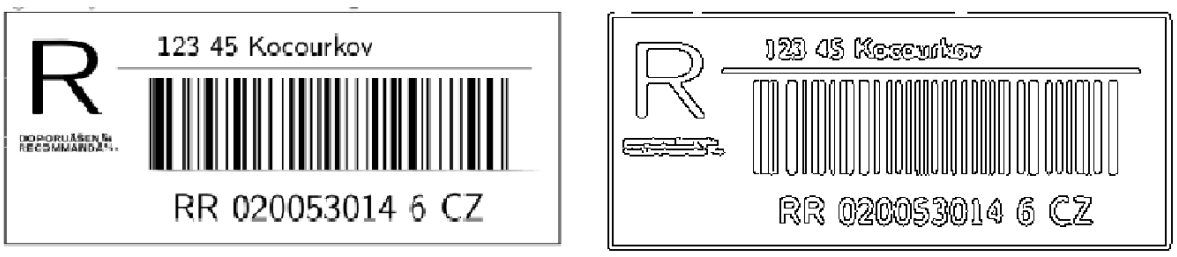
Nejjednodušší operátory diference vždy pouze v jednom směru (H - horizontální, V - vertikální, S - šikmý) vypadají takto:

$$H = \begin{bmatrix} 1 & -1 \end{bmatrix}; \qquad V = \begin{bmatrix} 1 \\ -1 \end{bmatrix}; \qquad S = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{4.4}$$

Jiné operátory jsou například:

-operátor Prewittové [11]:

$$h = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}; \qquad h = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}; \qquad h = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.5}$$



Obrázek 4.1: Originál (vlevo) a výsledek Cannyho hranového detektoru (vpravo). Byla použita funkce z programu Matlab.

-Sobelův operátor [11]:

$$h = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}; \qquad h = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}; \qquad h = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.6}$$

Operátorů je více, ale tyto jsou nepoužívanější při hranových detektorech. Samozřejmostí je možnost volby vlastního operátoru.

4.1.4. Cannyho detektor hran ve snímků

Jedná se o jeden z typů detektorů hran a nyní bude popsán jeho algoritmus.

Nejprve se určí první derivace, nejčastěji pomocí Sobelova operátoru 4.6. Určuje se ve všech čtyřech směrech, horizontální, vertikální a oba diagonální. Předtím je vhodné odstranit šum, aby nedocházelo k velkým změnám na pixelech obsahujících šum. Proto se používá Gausovo vyhlazení.

Výsledky hranových operátorů se považují za vektory (směrové pixely) a skládají se vektorově. Výsledek je opět vektor s úhlem natočení (tento úhel se většinou mění na čtyři základní směry).

Dochází ke ztenčení hran. Aplikuje se nemaximální suprese. Vyhodnocují se pixely zleva a zprava vyhodnocovaného směrového pixelu. Pokud je jeden z nich větší, pixel se vymaže.

Následuje prahování hran s hysterezí. Hystereze znamená, že hrany s intenzitou nad horním prahem jsou uznány ihned za hrany, hrany s intenzitou pod dolním prahem jsou ihned zavrhnuty a hrany s intenzitou mezi hranicemi jsou uznány za hranu, pokud byl uznán za hranu jich soused. Proces se opakuje, dokud ještě dochází ke změnám. [15].

Výsledek Cannyho hranového filtru je na obrázku 4.1.

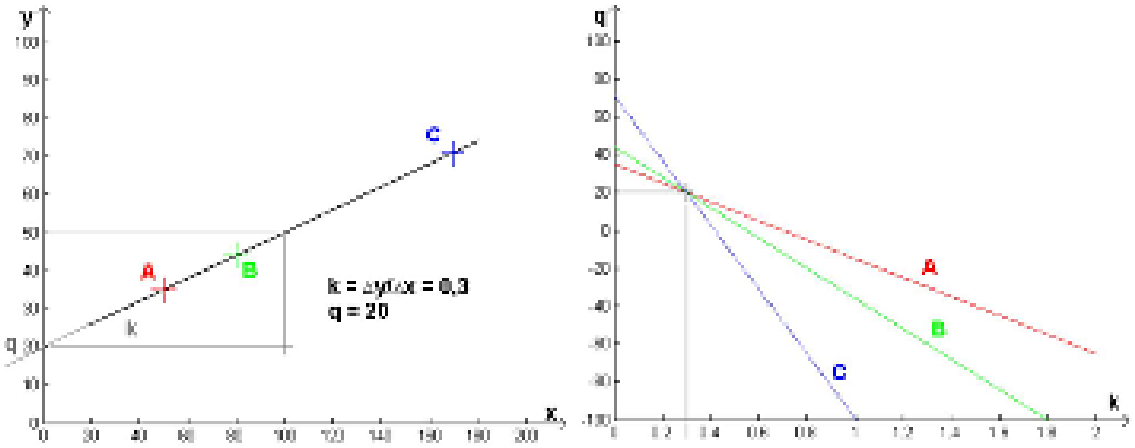
4.1.5. Houghova transformace

Tato transformace je definována pro hledání geometrických tvarů se známým analytickým popisem pomocí rovnice (přímka, elipsa, parabola...) [18].

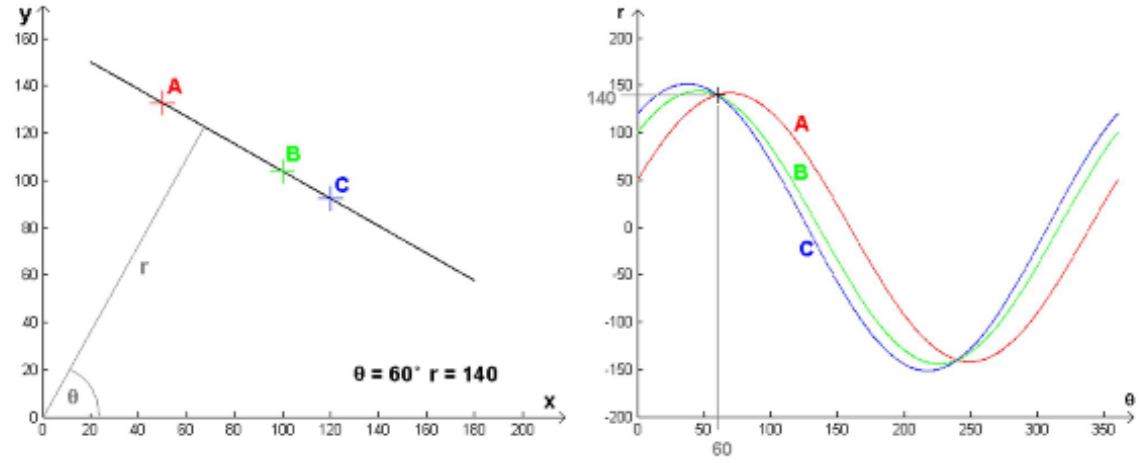
V případě 1-D a 2-D kódů bude nejdůležitější hledání přímky. Přímka se dá definovat jako:

$$y = kx + q \tag{4.7}$$

kde k je sklon a q je posun.



Obrázek 4.2: Ukázka transformace bodu na přímku a přímky na bod. [15]



Obrázek 4.3: Ukázka transformace bodu na křivku a přímky na bod. [15]

Houghova transformace najde bod obrazu náležící hraně a převede ho na přímku. Například bod v obraze $[x, y] = [10, 12]$. Dosadí se do rovnice 4.7: $12 = 10k + q$.

Po transformaci vznikne $q = -10k + 12$ a toto je rovnice bodu v Houghově transformaci.

Přímka se transformuje na bod. Z toho lze vyvodit, že body ležící na jedné přímce utvoří v transformaci přímky se společným průsečíkem. Toto nejlépe ilustruje obrázek 4.2. [15]

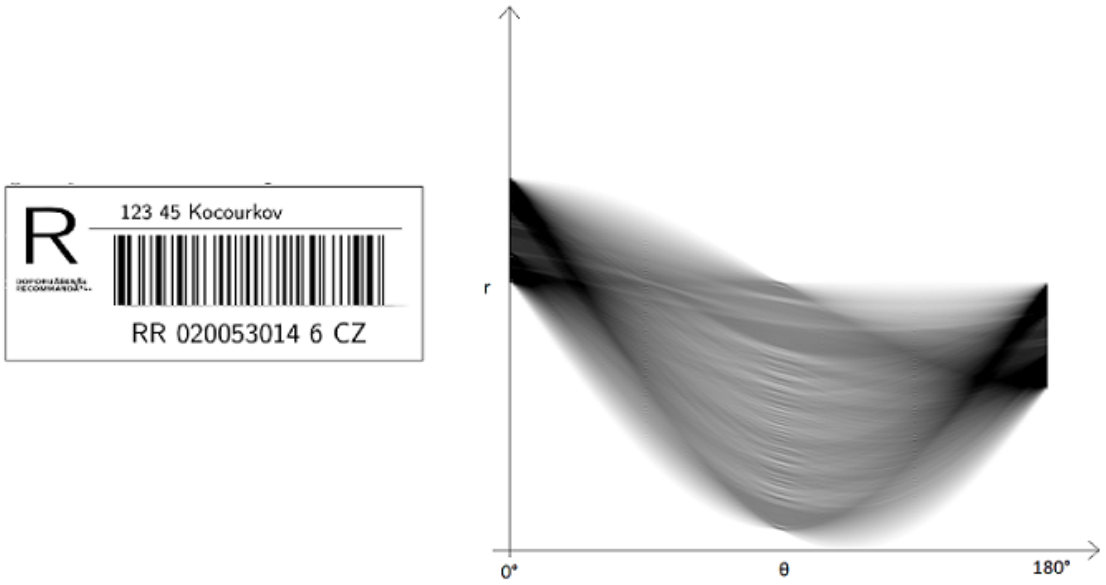
Poněkud zajímavější je druhá rovnice přímky, která využívá úhel náklonu mezi přímkou a osou x.

$$r = x \cdot cos\theta + y \cdot sin\theta \tag{4.8}$$

kde θ je úhel náklonu přímky k ose x a r je nejkratší vzdálenost přímky ke středu souřadnicové soustavy.

Zde se bod transformuje na křivku, ale přímka se nadále transformuje na bod (Obrázek 4.3).

Tento způsob je vhodnější pro použití u hledání kódů v obraze. Jelikož většina kódů má jeden (čárové kódy) nebo dva (QR-kód) dominantní směry natočení hran, můžeme z Houghovy transformace přímo určit dominantní úhel.



Obrázek 4.4: Výsledek Houghovy transformace (vpravo) snímku s kódem (vlevo).

Příklad Houghovy transformace je na obrázku 4.4. Z obrázku je patrné, že nejvíce bodů se seskupilo kolem 0° a 180°. To odpovídá skutečnosti, jelikož byla použita funkce v Matlabu, která jako osu x ve snímku používá kolmou osu, a tak pruhy kódu mají náklon 0°.

4.1.6. Fourierova transformace

Fourierova transformace provede rozklad signálu na jeho vyjádření pomocí sinusových signálů. [21] Toto je vhodné při filtraci šumu, jestli víme, jaké frekvence má šum nebo obraz.

V obraze si lze jako šum představit náhodné pixely, které mají vlastní frekvence a po Fourierově transformaci lze některé frekvence odstranit a tím odstranit šum, ale pokud jsou frekvence šumu podobné frekvencím objektů v obraze, nelze tyto frekvence filtrovat.

Transformace je definována rovnicí:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \tag{4.9}$$

Inverzní Fourierova transformace je definována jako:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{-i\omega t} d\omega \tag{4.10}$$

Diskrétní Fourierova transformace je definována jako suma:

$$D(n) = \sum_{k=0}^{N-1} d(k) e^{-in2\pi/N} \tag{4.11}$$

Inverze diskrétní Fourierovy transformace je pak:

$$d(k) = \frac{1}{N} \sum_{n=0}^{N-1} D(n) e^{in2\pi/N} \tag{4.12}$$

$f(t)$ je funkce v čase,
$F(\omega)$ je obraz $f(t)$ do frekvenční oblasti,
$d(k)$ je diskretizována $f(t)$
a $D(n)$ je obraz $d(k)$ ve frekvenční oblasti.

4.1.7. Dvourozměrná Fourierova transformace

Jedná se o rozšíření běžné jednorozměrné Fouriérovy transformace na další rozměr.

Signál (obraz) je převeden na reprezentaci pomocí prostorových sinusových frekvencí v obraze. Tuto frekvenci si lze představit jako vlnitý plech.

Výsledkem jsou komplexní čísla obsahující amplitudu a fázi. [17]

Pro zpracování obrazu bude používána diskrétní Fouriérova transformace 2D:

$$F(u,v) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)exp[-2\pi j(\frac{mu}{M}+\frac{nv}{N})] \tag{4.13}$$

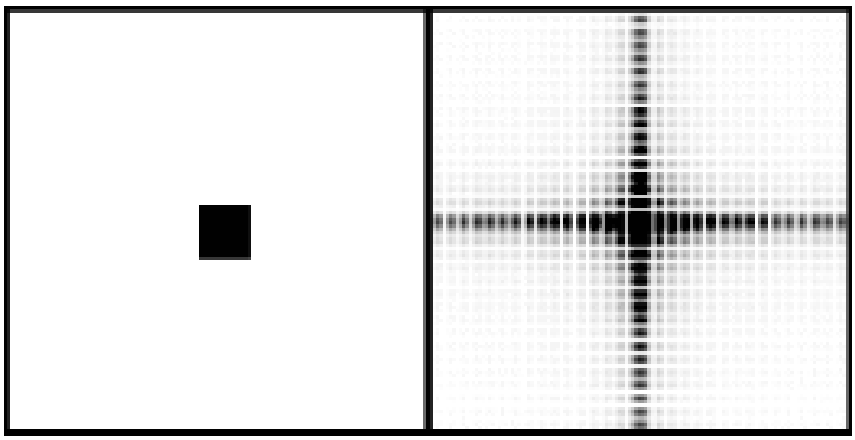
Inverzní transformace je potom:

$$\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} F(u,v)exp[2\pi j(\frac{mu}{M}+\frac{nv}{N})] \tag{4.14}$$

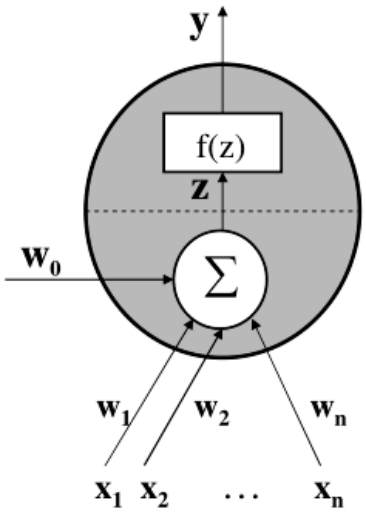
Z komplexní matice lze pak vyčíst čtyři informace:

1. amplitudu - absolutní hodnota čísel v matici,
2. fázi - dána komplexním číslem,
3. frekvence - vzdálenost od středu,
4. směr - poloha bodu vzhledem středu.

Vše ilustruje obrázek 4.5, kde je patrné, že čtverec se skládá hlavně z prostorových frekvencí kolmých na stěny čtverce.



Obrázek 4.5: 2D Fourierova transformace vstupního obrazu. [17]



Obrázek 4.6: Schéma neuronu. [13]

4.1.8. Neuronová síť

Neuronová síť bude vysvětlena, ale je možné, že nedojde k jejímu použití v konečném řešení. Každopádně její využití skýtá obrovské možnosti. Například článek *Robust Angle Invariant 1D Barcode Detection* [33] popisuje její použití při vyhodnocení výsledků Houghovy transformace. Toto bude ještě blíže vysvětleno dále.

Neuronová síť sestává z neuronů.

Neuron má vstupy (x_k) s váhami(w_k), práh (w_0), přenosovou funkci ($f(z)$) a výstup (y). Vše je znázorněno na obrázku 4.6.

Přenosová funkce $f(z)$ může být skoková, bipolární, hyperbolická tangenta... Výstup potom bude:

$$y = f(w_0 + \sum_{i=1}^{N} x_i w_i) \tag{4.15}$$

V praxi se používá síť výše popsaných neuronů, která se skládá z výstupní vrstvy, vnitřních vrstev a vrstvy vstupní, která je často tvořena jen neurony s lineárním přenosem, jedním vstupem s váhou 1 a bez prahu. Vstupní vrstva potom slouží jen k distribuci vstupů do sítě.

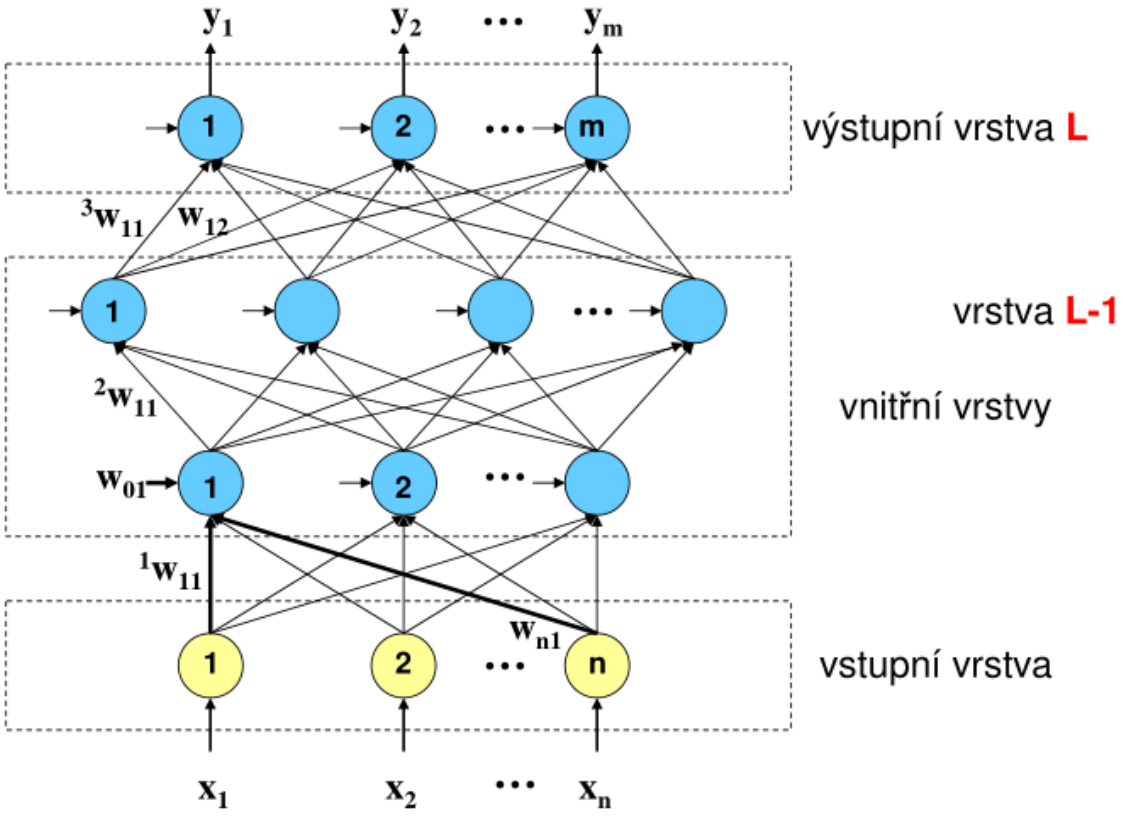
Struktura běžné neuronové sítě je znázorněna na obrázku 4.7.

Učící algoritmus backpropagation

Existuje celá řada učících algoritmů. Zde bude popsán pouze algoritmus backpropagation neboli algoritmus zpětného šíření chyby.

Tento algoritmus tvoří čtyři kroky:

1. dopředné šíření vstupního signálu,
2. výpočet chyby,
3. zpětné šíření chyby,
4. adaptace vah (na počátku se váhy volí náhodně).



Obrázek 4.7: Struktura vícevrstvé neuronové sítě. [14]

Adaptace vah se provede dle vzorce:

$$w_{jk}(t+1) = w_{jk}(t) + \Delta w_{jk}(t+1) \tag{4.16}$$

kde j je výchozí neuron a k je koncový neuron.

a kde

$$\Delta w_{jk} = \alpha * \frac{\sigma f(z)}{\sigma z} ex \tag{4.17}$$

slovně: koeficient učení $\times$ derivace přenosové funkce $\times$ chyba neuronu (e) $\times$ vstup neuronu (x).

Občas se přidává i moment (M), který zrychluje učení:

$$\Delta w_{jk} = \alpha \frac{\sigma f(z)}{\sigma z} ex + M(w_{jk}(t) - w_{jk}(t-1)) \tag{4.18}$$

Chyba neuronu e se vypočte pro výstupní vrstvu jako:

$$e = d - y \tag{4.19}$$

kde d je žádaný výstup a y je výstup sítě.

Pro vnitřní vrstvy je výpočet následující:

$$e_{ij} = \sum_{k=1}^{n} w_{ijk} e_{i+1} \tag{4.20}$$

kde n je počet vah v neuronu.

Jinak lze říci, že chyba j-tého neuronu v i-té vrstvě je rovna sumě chyb neuronů ve vyšší vrstvě vynásobené odpovídajícími váhami. Chyba se pohybuje v opačném směru jako data a stejně jako data je násobená váhou, když prochází spojem. Chyby, které se potkají v neuronu, se sčítají.

Proces učení se přeruší ukončovací podmínkou. Buďto se dosáhne maximálního počtu iterací, nebo chyba sítě klesne pod nastavenou hranici.

Chyba sítě E_c je suma chyb E_h, to je chyb vzhledem ke všem vzorům.

$$E_c = \sum_{h=1}^{p} E_H = \frac{1}{2} \sum_{h=1}^{p} \sum_{j=1}^{m} (d_{hj} - y_{hj})^2 \tag{4.21}$$

kde p je počet vzorů a m je počet neuronů.

Vybavování

Vybavování začíná přivedením dat na vstup.

Data se šíří sítí, při přechodu z vrstvy do vrstvy se násobí odpovídajícími váhami a nakonec projdou až k výstupní vrstvě.

Z výstupní vrstvy vychází výsledek.

4.2. Třídicí stroj na dopisy

Pro pochopení IMB kódu je potřeba pochopit jeho použití a tím je primárně třídění poštovních zásilek v obálkách (dopisy).

Přístup ke stroji a popis jeho funkce byl zprostředkován pracovníky brněnské třídírny České pošty a.s.

Jedná se o masivní třídicí stroj vyrobený a dodaný firmou Siemens.

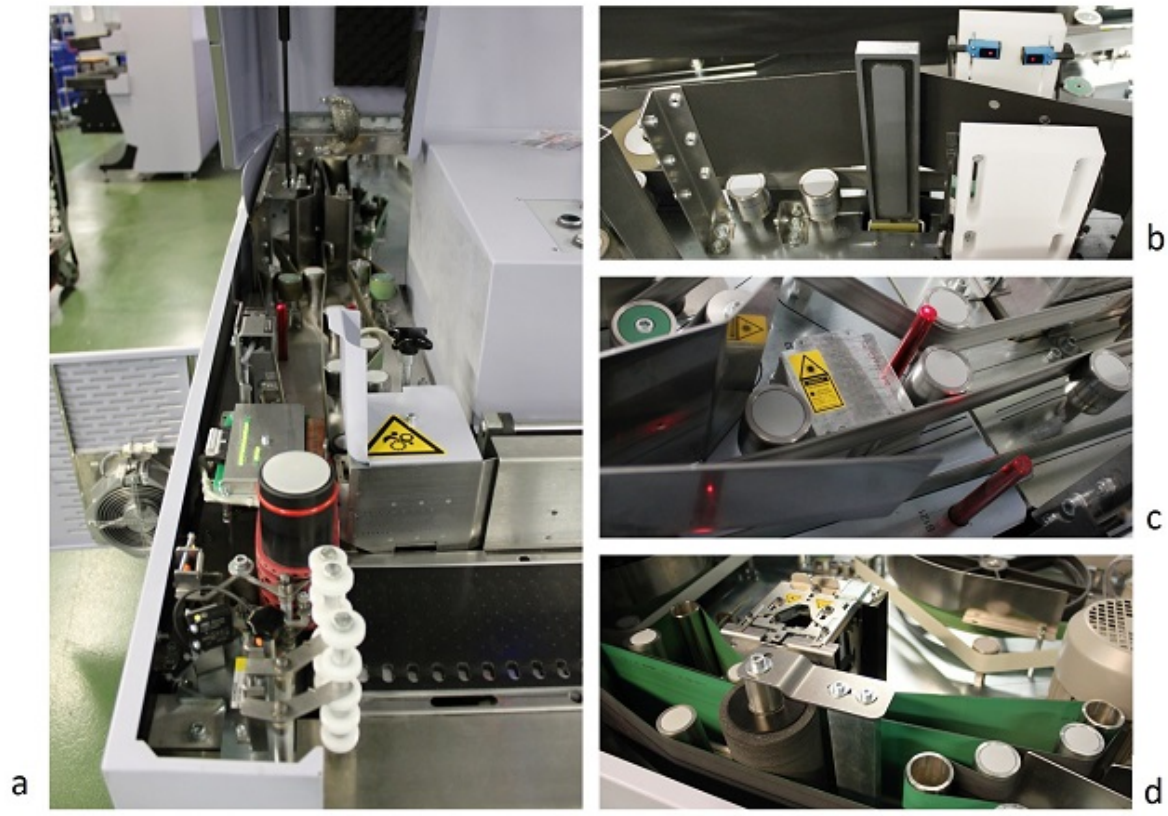
Obálky začínají svou pouť u vstupu do stroje, kde jsou přisány na dopravník vývěvou a dále pásy posouvány do stroje. (Obrázek 4.8-a).

Prvně se kontroluje výška obálky. Příliš vysoké strojem neprojdou. Stejně tak se detektorem kovů kontroluje přítomnost kovových součástí, které by mohly poškodit stroj. (Dle vyjádření obsluhy je velmi populární zasílání klíčů a paměťových klíčenek USB.) Také se kontroluje pružnost zásilky, protože ve stroji musí projít několika zatáčkami a při nízké ohebnosti by mohlo dojít k jejímu poškození. (Obrázek 4.8-b)

Dále následuje kontrola přítomnosti IMB kódu. Ten je natištěn na spodní části obálky barvou citlivou na ultrafialové světlo (toto bude podstatné později). Pokud je kód nalezen, je přečten a neaktivuje se kamera s OCR adresy. (Obrázek 4.8-c)

V případě, že není nalezen IMB kód, spustí se kamera, která pořídí snímek a program provede rozpoznání adresy. Pokud není možné strojové přečtení adresy, snímek se odešle do počítačového centra na obrazovku příslušného pracovníka, který provede ruční zadání směrovacího čísla nebo celé adresy. Aby se vše stihlo, je zařazena zpožďovací linka - u starších strojů 10 sekund, u novějších 15 sekund. (Obrázek 4.8-d)

Pokud nebyl na obálce IMB kód, tak se na obálku vytiskne pomocí tiskárny barvou citlivou na ultrafialové záření. Na obrázku 4.9-c je vidět tiskárna v odstávce. Při provozu je namontována na stojan a směřuje směrem na pás.



Obrázek 4.8: Vnitřek třídicího stroje. a) Přehledový pohled na podavač obálek a vstupní část. b) Kontrola rozměrů a detektor kovů. c) Čtečka IMB kódu. d) Kamera s OCR adresy.

Tiskárna také natiskne číslo, které se skládá z čísla třídicího stroje (pořadí určuje pořadí montáže v České republice), data natištění a poštovního směrovacího čísla v adrese příjemce.

Veškerá práce s kódem zásilky zakódovaným IMB kódem se odehrává na serverech (Obrázek 4.9-b). Tyto servery jsou napojené na databázi všech kódů v Praze a můžou identifikovat cestu zásilky, typ zásilky a jiné podstatné informace.

Posledním krokem je zařazení zásilky do příslušné přihrádky (Obrázek 4.9-a). Obálky vyřazené v některém z kroků padají do speciální přihrádky, ze které jsou vyzvednuty a roztříděny ručně. Podle obsluhy je drtivá většina obálek se správně napsanou adresou správně zařazena.

Jako příklad důležitosti správného formátu adresy lze uvést, že při ukázce funkce stroje byly použity testovací obálky, které měly poštovní směrovací číslo umístěné nad názvem města. Tyto obálky byly zařazeny správně jen z ±20%. Není známo, proč se jako testovací používají zrovna tyto špatně popsané obálky, ale v běžném provozu stroj zavrhne pár desítek zásilek na tisíce roztříděných.

Obrázek 4.9: Třídicí stroj. a) Pohled na část s přihrádkami na roztříděné obálky. b) Server. c) Tiskárna IMB kódu v odstávce.

4.3. Přehled metod z různých vědeckých článků

Tato kapitola se pokusí poskytnout přehled současných metod hledání kódů ve snímku na základě vědeckých článků.

4.3.1. Robust Angle Invariant 1D Barcode Detection

Robustní, nezávislá na úhlu detekce 1D čárového kódu.

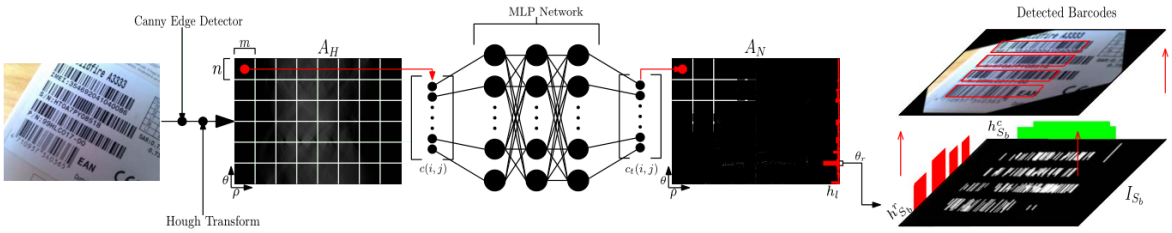
Autoři navrhují zajímavé řešení s použitím neuronové sítě. Metoda má úspěšnost 85% a je náročná na vstupní data při určení sítě.

Algoritmus:

1. Výpočte se Houghova transformace vstupního obrazu.
2. Matice Houghovy transformace A se souřadnicemi r a ρ se rozdělí na matice c o velikosti $m \times n$.
3. Matice c se vloží na vstup neuronové sítě.
4. Na výstupu neuronové sítě je pak matice c_t, ve které mají pixely, o kterých si neuronová síť myslí, že reprezentují čáru náležící čárovému kódu, hodnotu jedna.
5. Z matic c_t se opět sestaví celková matice A_N.

6. Sečtou se hodnoty v řádcích odpovídajících jednomu úhlu.
7. Vybere se maximum a odečte se souřadnice maxima, která značí úhel natočení obrazu. Provede se rotace o tento úhel.
8. Následně se na základě kombinace dat z matice A_N a vlastností textur vybere oblasti s čárovými kódy.
9. Segmentace na jednotlivé kód probíhá pomocí sumace řádků a sloupců

Celý algoritmus je znázorněn na obrázku 4.10



Obrázek 4.10: Znázornění algoritmu z článku Robust Angle Invariant 1D Barcode Detection. [33]

Zhodnocení použitelnosti metody pro tuto práci

Algoritmus nezvládá čtení kódů, které nemají stejnou orientaci. Je také náročný na počáteční data pro určení neuronové sítě, kterých musí být mnoho a jsou vytvářeny ručním přiřazováním obrazových bodů c_t obrazovým bodům c.

Výhodou je, že kód může být částečně poškozený.

Metoda nedává až tak dobré výsledky. Zdá se být zbytečné používat neuronovou síť, když potřebná data pro natočení snímku se dají vyčíst přímo z matice Houghovy transformace.

Výsledky neuronové sítě sice pomáhají určit polohu kódu, ale i na to existují jednodušší metody. Zvláště, že oproti jiným metodám má tato metoda velmi nízkou úspěšnost.

4.3.2. Application of Computational Verb Image Processing to 1-D Barcode Localization

Aplikace spočetných sloves pro zpracování obrazu při hledání 1D čárového kódu. [12]

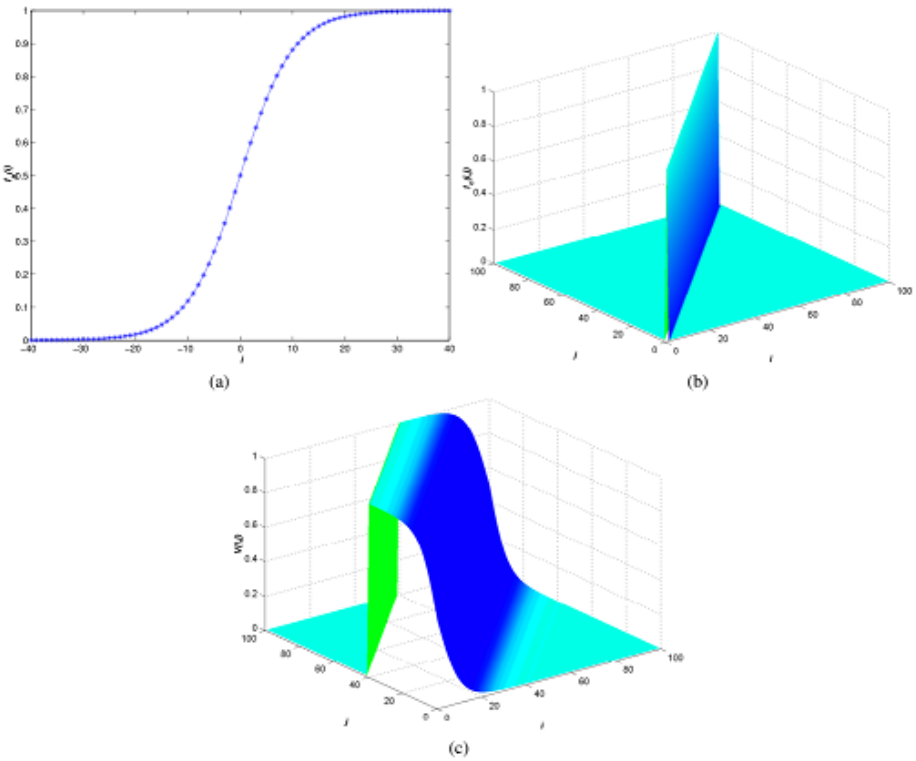
Jedná se o metodu používající prostorová spočetná slovesa (*spatial computational verbs*). Jde o matematický nástroj, který je schopný vyjádřit změnu v prostoru. Ve zpracování obrazu se jedná o vyjádření jasu v propojení se souřadnicemi.

Prostorovým slovesem je například postupné snižování jasu za hranou nebo jasová změna na šikmé střeše.

Funkce prostorového slovesa ν pro snímky se definuje takto:

$$\epsilon_\nu : \Omega_S \rightarrow \Omega_B \tag{4.22}$$

Ω_S je podpora(*support*) pro dvoudimenzionální snímek a Ω_B je rozsah jasu (odstínů šedi).



Obrázek 4.11: Proces generování kanonického prostorového slovesa. a) f_p funkce profilu jasu. b) f_o funkce tvaru obrysu. c) Vyvíjející se funkce $v(i,j)$ výsledného kanonického prostorového slovesa. [24]

Pro konstrukci kanonického prostorového slovesa se použije funkce profilu jasu f_p a funkce obrysu tvaru f_o.

Vyvíjející se funkce ϵ_ν se potom vypočte takto:

Řádková kompozice

$$\epsilon_\nu(i,j) = \oplus_{k=-\infty}^{\infty} \oplus_{l=-\infty}^{\infty} f_p(k,l) \otimes f_o(i-k, j-l) \tag{4.23}$$

kde $\oplus$ je s-norma (maximum) a $\otimes$ je t-norma (minimum).

Pro ulehčení výpočtu pro běžné použití se výpočet rozkládá do dvou směrů:

řádková kompozice

$$\epsilon_\nu(i,j) = \oplus_{l=-\infty}^{\infty} f_p(l) \otimes f_o(i, j-l) \tag{4.24}$$

sloupcová kompozice

$$\epsilon_\nu(i,j) = \oplus_{k=-\infty}^{\infty} f_p(k) \otimes f_o(i-k, j) \tag{4.25}$$

Řádkový výpočet ilustruje obrázek 4.11.

Podobnost dvou sloves se vypočte podle rovnice 4.26

$$S(v_1, v_2) = \begin{cases} 1 - \frac{\sum_{(i,j)\in\Omega_S} |v_1(i,j) - v_2(i,j)|}{\sum_{(i,j)\in\Omega_S} v_1(i,j) + v_2(i,j)} & \text{když} \sum_{(i,j)\in\Omega_S} v_1(i,j) - v_2(i,j) \neq 0 \\ 0 & \text{když} \sum_{(i,j)\in\Omega_S} v_1(i,j) - v_2(i,j) = 0 \end{cases} \tag{4.26}$$

kde v_1 a v_2 jsou vyvíjející se funkce prostorového slovesa. [24]

Pro hledání vzoru ve snímku se jako jedno ze slov použije šablona, tzv. šablonové sloveso(*template verb*). Jeho předpis pro hledání čárového kódu je v rovnici 4.27.

$$\begin{aligned} &\text{funkce jasu} \\ f_p(i) &= (0.1, 1, 0.1) \\ &\text{funkce tvaru obrysu} \\ f_i(i.j) &= \begin{cases} 1 & \text{když } j\, mod 4 == 0 \\ 0 & \text{jinak} \end{cases} \\ f_i(i.j) &= \begin{cases} 1 & \text{když } i\, mod 4 == 0 \\ 0 & \text{jinak} \end{cases} \end{aligned} \tag{4.27}$$

Algoritmus hledání čárového kódu navržený autory článku [12] je následující:

1. Snímek se převede do odstínů šedi a aplikuje se Cannyho hranový detektor.
2. Výpočte se šablonové sloveso ν_1 s rozměry podpory $p \times q$ (31×31), jedno pro řádky a jedno pro sloupce podle rovnic 4.24, 4.25 a 4.27.
3. Vybere se kotevní bod tak, aby byl ve středu okna o velikosti $p \times q$ (31×31).
4. Vypočte se vyvíjející se funkce slovesa ν_2 pro zvolené okno a kotevní bod. Opět jak pro řádky, tak pro sloupce. Jako podpora se použije zvolené okno ze snímku a jako funkce vývoje jasu se použije stejná funkce f_p jako pro šablonové sloveso.
5. Porovnají se obě slovesa a výsledek se zapíše na souřadnice kotevního bodu.
6. Výsledkem jsou dvě matice řádkového a sloupcového výpočtu. Vypočte se suma prvků obou matic. Pokud je suma matice řádkového výpočtu větší, je kód orientovaný spíše svisle a dále se použije tato matice. Opačný případ je obdobný.
7. Zvolí se pouze body s velkou hodnotou podobnosti, protože s největší pravděpodobností reprezentují oblast kódu.
8. Někdy text a jiné prvky můžou rovněž vykazovat velkou podobnost se šablonou, proto se provede výpočet úhlu všech bodů v matici podobnosti na základě jejich sousedství. Vyberou s pouze ty, které reprezentují rovnoběžné čáry.
9. Nakonec se opíše nejmenší možný obdélník, který obsahuje všechny body matice porovnání a toto by měla být hranice kódu.

Výhodou algoritmu je úspěšnost 96.40 % uváděná autory. Dále schopnost najít kód ve špatných světelných podmínkách a při deformaci. Tento algoritmus lze upravit i pro hledání více kódů v jednom snímku.

Tento popis plně nevysvětluje podstatu použití prostorových spočetných sloves. Pro lepší pochopení se doporučuje literatura [24].

Zhodnocení použitelnosti metody pro tuto práci

Algoritmus byl implementován v prostředí Matlab pouze do kroku nalezení bodů s vysokou hodnotou podobnosti (7. bod algoritmu). Bylo snahou zjistit chování algoritmu. Jako vstup byl použitý snímek čárového kódu EAN-13. Snímek byl pořízen mobilním zařízením Nokia 620.

Výsledky byly docela uspokojující. Problémem je správné určení prahu filtrace bodů s vysokou podobností v 7. bodě.

Šablona navrhována autory článku úspěšně našla pouze paralelní tenké čáry o vzdálenosti v násobcích 4. Pro případné další použití by bylo vhodné zvážit změnu šablony pouze na jednu čáru, ale tak, aby byla zachována její dostatečná velikost.

Byla vyzkoušena šablona 7×7 s čárou uprostřed a tato šablona, zdá se, dávala lepší výsledky.

Po zhodnocení algoritmu bylo jasné, že podobného výsledku by bylo dosaženo při vynechání výpočtu prostorové funkce slovesa jak vstupního snímku, tak šablony. Toto bylo vyzkoušeno a bylo aplikováno pouze porovnávání se šablonou.

Výsledkem bylo zjištění, že aplikace prostorového slovesa výrazně zlepšuje proces porovnávání díky zavedení jakéhosi okolí.

4.3.3. Improving Barcode Detection with Combination of Simple Detectors

Vylepšení detekce čárových kódů kombinací jednoduchých detektorů. Tento článek popisuje pár metod detekce kódu ve snímku a pak jejich kombinaci. [3]

Na počátku se doporučuje provést filtraci snímku Gausovým filtrem, mediánovým filtrem nebo podvzorkování pro zlepšení šumových vlastností snímku.

Dále budou popsány jednotlivé metody zmíněné v článku.

Použití Houghovy transformace pro detekci čar

Aplikuje se Cannyho detektor hran a pravděpodobnostní Houghova transformace. Jedná se o obdobu Houghovy transformace, kdy se určí minimální délka čáry a maximální délka mezery mezi dvěma body, kdy jsou shledány jako součást jedné přímky. Výstupem je vektor úseček určený koncovými body.

Bohužel je nutné znát přibližné rozměry kódu pro určení parametrů transformace.

Výsledkem by měl být seznam úseček tvořený jejich centrálními body, délkou a orientací. Následně se provede shlukování na základě těchto parametrů a vyberou se shluky s dostatečným počtem úseček. Tedy i zde je nutné znát přibližný vzhled kódu.

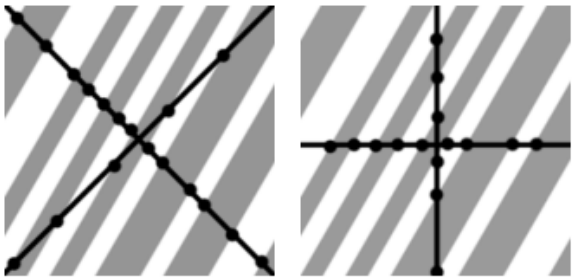
Z výsledku lze určit oblast kódu.

Jednotné dělení pomocí distanční transformace

V první řadě je nutné rozdělit snímek na menší části (bloky). Bloky s podobnými texturami mají podobné lokální statistické vlastnosti. Proto se hledají bloky, které reprezentují kód s velkou pravděpodobností.

Vypočte se distanční mapa. Distanční mapa se počítá z binárního obrazu, zde z Cannyho detektoru hran, a hodnota odpovídá vzdálenosti obrazového bodu k nejbližšímu nenulovému obrazovému bodu. [7]

Obrázek 4.12: Metoda distanční transformace. a) Originální snímek, b) Detektor hran. c) Distanční mapa (hodnoty upraveny pro vizualizaci). [3]



Obrázek 4.13: Ukázka hledání statistiky kontrastu.

Pro každý blok distanční mapy se vypočte střední hodnota a standardní odchylka. Pro 1D kódy je hodnota mediánu rozložena mezi polovinu minimální a polovinu maximální šířky čar.

Po výpočtu vlastností všech bloků hledáme shluky v matici vlastností, která byla vyprahována. Samozřejmě i zde je předpoklad znalosti rozměrů kódu.

Nakonec se malé shluky zahodí a velké se vyhodnotí jako oblasti kódu.

Měření kontrastu s jednotným dělením

Opět se provede dělení na jednotné bloky. Základem je vlastnost jednodimenzionálních čárových kódů, které mají vysoký počet změn jasu v jednom směru.

Jednotlivými čtverečky se vedou dvě dvojice přímek (znázorněno na obrázku 4.13). Počítá se počet hran na jednotlivých přímkách. Hodnota čtverečku se pak určí jako maximum z dvou hodnot, jedné pro každý pár. Hodnota pro jeden pár se počítá podle vzorce 4.28.

$$C_i = \frac{|V_{i1} - V_{i2}|}{max(V_{i1}, V_{i2})} \tag{4.28}$$

kde V_{ij} je počet hran na přímce j páru i.

Nakonec se opět provede prahování matice vlastností, shlukování a hledání velkých shluků.

Konečný algoritmus doporučovaný autory článku

1. Vstupní filtrování
2. Cannyho hranový detektor

3. Distanční transformace
4. Min-max transformace
5. Pravděpodobnostní Houghova transformace a určení pravděpodobnosti, že bod náleží kódu.
6. Rozdělení na čtverečky
7. Výpočet pro každý čtvereček střední hodnoty vzdálenosti, kontrastních vlastností a lokálního shlukování
8. Filtrace hodnot z předešlého bodu
9. Provedení sloučení pomocí hlasování, vyhrává majoritní hlas nebo maximální hodnota, případně se provede váhované hlasování.

Zhodnocení použitelnosti metody pro tuto práci

Bohužel se nepodařilo získat přístup k podrobnějšímu popisu. Některé části, jako min-max transformace a lokální shlukování, jsou popsány v jiných článcích, které nejsou veřejně přístupné. Přesto může být tento algoritmus vodítkem ke tvorbě vlastního algoritmu.

Popsané jednotlivé metody jsou jednoduché, ale bohužel závislé na znalosti přibližných rozměrů kódu ve snímku. Přesto jsou některé z těchto metod vhodné pro použití. Metoda shlukování v matici vlastností je rozhodně dobrá pro koncové hledání, problémem je sestavení vhodné matice vlastností.

4.3.4. Reading barcodes using digital cameras through image processing

Čtení čárových kódů při použití digitálních fotoaparátů pomocí zpracování obrazu. [1]

Jedná se o starší článek, který si kladl za úkol urychlit čtení produktu na pokladnách v obchodech.

Zpracovává se snímek v odstínech šedi. Jako první se provede Canynho detekce hran s jádrem pouze v kolmém směru (očekává se přibližná kolmost hran kódu).

Po detekci hran se sestaví tabulky sousedství v podobě polí. Prochází se postupně body hran. Pokud existuje soused, který je už v některé tabulce, přidá se do této tabulky, pokud soused neexistuje, vytvoří se nová tabulka.

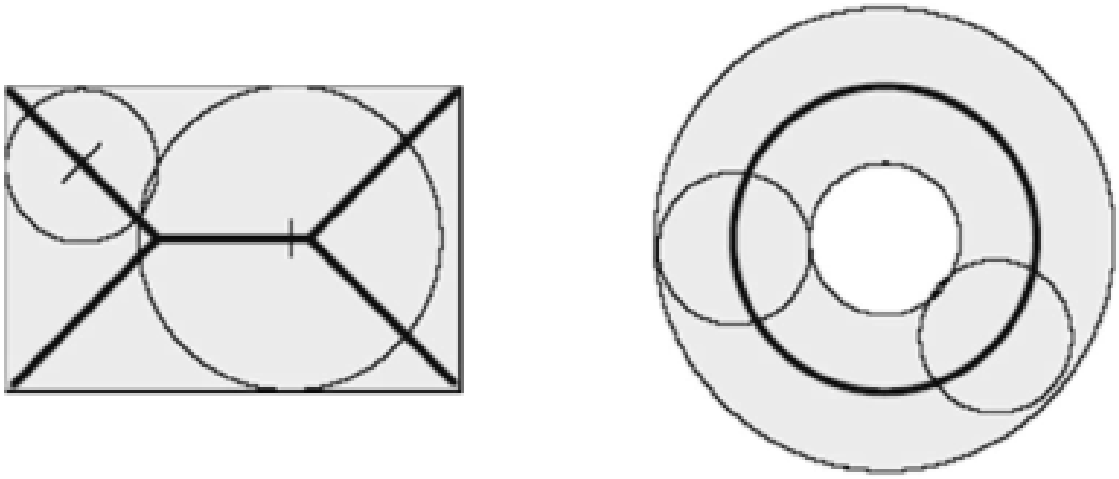
Myšlenka je taková, že tabulky obsahující hrany kódu jsou přibližně stejně dlouhé. Počítá se standardní odchylka délek. Vyberou se ty, které mají malou odchylku. Algoritmus počítá s kódem EAN-13, kde známe délku a počet čar, proto lze algoritmus vylepšit hledáním pouze nějakého, blíže neurčeného počtu nejlepších kandidátů.

Seznam vybraných tabulek obsahuje souřadnice hran kódu. Z těchto souřadnic už lze určit oblast kódu.

Zhodnocení použitelnosti metody pro tuto práci

Algoritmus vyžaduje velký počet prohledávání tabulek sousedství a předpokládá alespoň minimální znalost velikosti kódu. Pro použití s GS1-128 by se nutnost této znalosti dala eliminovat.

Obrázek 4.14: a) Blok po Otsu prahování. b) Blok po skeletonizaci. [4]



Obrázek 4.15: Princip skeletonizace. [10]

Jelikož je to starší článek, jsou metody v něm použité poněkud zastaralé. Myšlenka není špatná, ale dnes by se asi použila pravděpodobnostní Houghova transformace a seznam úseček jí určený místo tabulek sousedství.

4.3.5. Locating and Decoding EAN-13 Barcodes from Images Captured by Digital Cameras

Lokalizace a detekce EAN-13 čárového kódu ve snímku pořízeném digitálním fotoaparátem. [4]

Opět se používá jen snímek v odstínech šedi. Na vstupu se snímek převzorkuje na velikost v násobcích 32, jelikož algoritmus pracuje po blocích 32×32 obrazových bodů.

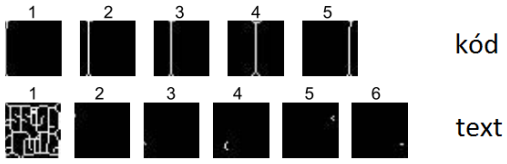
Detekce kódu

Obraz je rozdělen na nepřekrývající se bloky 32×32 obrazových bodů.

Na každý blok je aplikováno Otsu prahování a následně je na binární data aplikována morfologická operace skeletonizace (obrázek 4.14). Jedná se o postupné ztenčování až na šířku jednoho obrazového bodu. Pokud se do objektu vepíše kruh, který se dotýká hran objektu ve dvou bodech, pak střet tohoto kruhu tvoří bod skeletu (obrázek 4.15).

Dále jsou v každém bloku extrahovány spojité oblasti do samostatných bloků (obrázek 4.16).

Pro každý oddělený blok se spočítá úhel. Úhel se počítá mezi osou x a hlavní osou elipsy, která má stejný moment druhého řádu jako vybraný region. Blok s více paralelními



Obrázek 4.16: Rozdělení spojitých oblastí. [4]

spojitými oblastmi se vyhodnotí jako blok čárového kódu a jeho orientace se vypočte jako průměr úhlu v bloku.

Nakonec se vybere oblast bloků, které mají podobný úhel a jsou blízko sebe.

Moment druhého řádu, pomocí kterého se počítá úhel natočení bloku, se počítá následovně: [8].

Moment obrazu se počítá podle vzorce 4.29:

$$M_{ij} = \sum_{x=1}^{W} \sum_{y=1}^{H} x^i y^j f(x, y) \tag{4.29}$$

kde $f(x, y)$ reprezentuje obraz o šířce W a výšce H.

Potom těžiště binárního obrazu se vypočte jako:

$$(\bar{x}, \bar{y}) = (\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}) \tag{4.30}$$

Dále centrální moment kolem těžiště se vypočte jako:

$$\mu_{ij} = \sum_{x=1}^{W} \sum_{y=1}^{H} (x - \bar{x})^i (y - \bar{y})^j f(x, y) \tag{4.31}$$

Konečně se vypočte kovariantní matice:

$$\begin{bmatrix} \mu'_{20} & \mu'_{11} \\ \mu'_{11} & \mu'_{02} \end{bmatrix} = \begin{bmatrix} \mu_{20}/\mu_{00} & \mu_{11}/\mu_{00} \\ \mu_{11}/\mu_{00} & \mu_{02}/\mu_{00} \end{bmatrix} \tag{4.32}$$

a úhel natočení se určí jako:

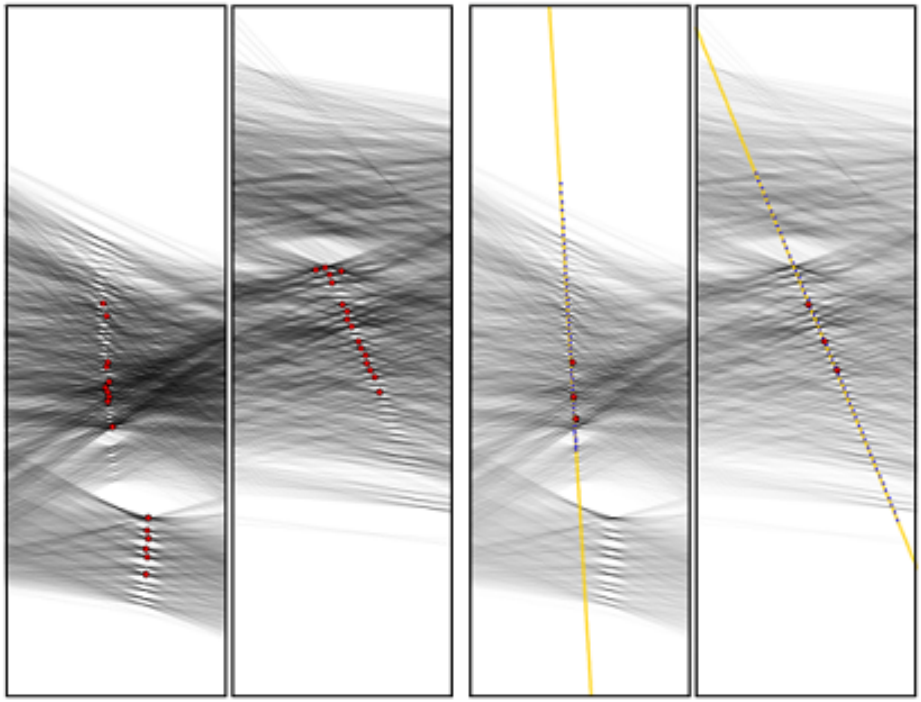
$$\Phi = \frac{1}{2} arctan(\frac{2\mu'_{11}}{\mu'_{20} - \mu'_{02}}) \tag{4.33}$$

Dekódování kódu

Přes kód je vedena přímka. Hodnoty jasu podél přímky se berou jako hodnoty funkce jasu podél přímky. Je provedeno prahování funkce jasu. Nejprve je zapotřebí najít startovací sekvenci a dále již není problém dekódovat zbytek.

Zhodnocení použitelnosti metody pro tuto práci

Základ metody je dobrý. Použití výpočtu úhlu natočení, ať už je použit skelet nebo hrany, malých části obrazu a jejich následná segmentace nebo shlukování by mohla být jednoduchá a efektivní metoda nalezení kódu.



Obrázek 4.17: a) Nalezena maxima v Hougových transformacích dvou dominantních skupin. b) Vygenerované hypotézy.

4.3.6. Fast Detection and Recognition of QR codes in High-Resolution Images

Rychlá detekce a rozpoznání QR-kódu ve snímcích s vysokým rozlišením. [19]

Detekce QR-kódu je založena na podobných principech jako detekce 1D čárových kódů. Samozřejmě se metody musí rozšířit o druhou dimenzi.

Například hledání pomocí paralelních hran bude muset být provedeno v obou směrech a bude se hledat místo s největším počtem paralelních a na sebe kolmých hran.

Zmiňovaný článek popisuje detekci QR-kódu na základě PClines a sestává ze čtyř kroků.

Nejprve se provede detekce hran a gradientu hran (úhlu natočení). Sestaví se histogram úhlů a vyberou se dva dominantní úhly ($\pm 90°$ vzdálené).

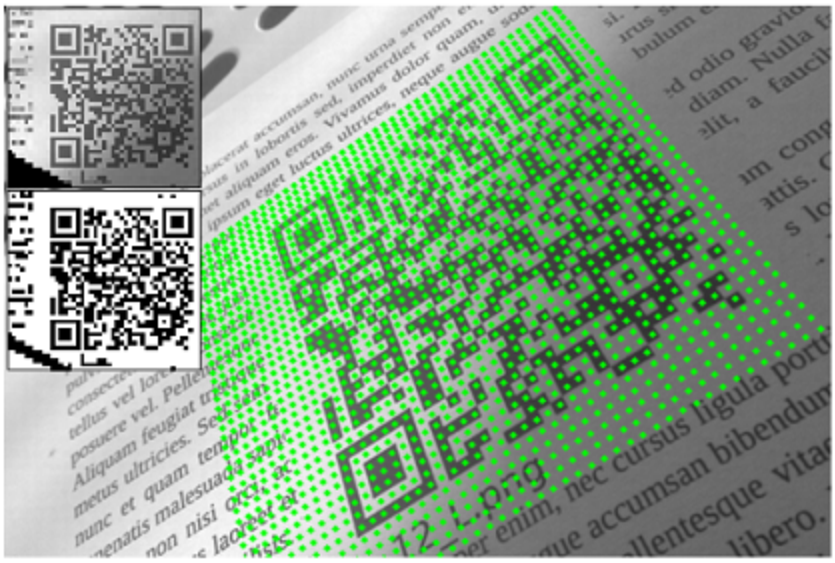
Za druhé je pro každou ze dvou dominantních skupin hran spočítána Houghova transformace.

Za třetí se najdou lokální maxima v matici Houghovy transformace. Tato maxima jsou použita pro vytvoření hypotézy o skupinách souběžných přímek. Hypotéza je definována třemi náhodnými body (obrázek 4.17). Pro každou hypotézu se vypočte skóre. Je to rozdíl mezi reálnými maximy a proloženými maximy. Hledá se minimální hypotéza.

Za čtvrté se provede extrakce bitmapy značek. Jako vzorkovací body se použijí body v polovině mezi dvěma maximy v Houghově transformaci (obrázek 4.18).

Detekce více kódů ve snímku

Snímek se nejprve rozdělí na menší bloky a u těch je určena pravděpodobnost, že je blok součástí QR-kódu.



Obrázek 4.18: Značky ve vzorkovacích místech. Po vzorkování je aplikované prahování.

Poté se vypočte histogram gradientů hran v bloku (obdobně jako bylo popsáno výše). Dále je pro každý blok sestaven vektor vlastností 4.34, který obsahuje normalizovaný histogram gradientů (h_{norm}), dva dominantní gradienty (α_1,α_2), počet hranových bodů (N_e) na celkový počet obrazových bodů (A) a pravděpodobnost přítomnosti šachovnicové mřížky (p).

$$v = (p, \alpha_1, \alpha_2, \frac{N_e}{A}, h_{norm}); \tag{4.34}$$

Pravděpodobnost se vypočte podle rovnice 4.35

$$p = \left(1 - \frac{||\alpha_1 - \alpha_2| - \frac{\pi}{2}}{\frac{\pi}{2}}\right) \frac{2min(h_a, h_b)}{h_a + h_b}; \alpha_1 < \alpha_2 \tag{4.35}$$

kde h_a a h_b jsou hodnoty v histogramu, které odpovídají dominantním gradientům.

První část říká, že úhly jsou vzdálené 90°, a druhá, že se preferuje ostrý vrchol v histogramu.

Nyní se provede segmentace bloků do skupin S metodou jednoduchého 4okolního barvení (*simple 4-neighborhood blob coloring*). Použijí se pouze hodnoty α_1, p a $\frac{N_e}{A}$.

Dále se provede segmentace bloků. Skóre pravděpodobnosti, že je blok součástí QR-kódu, se vypočte podle rovnice 4.36:

$$P(S_i) = \frac{1}{A(S_i)} \sum_{\tau \in S_i} p\frac{N_e}{A}, i \in 1, 2, ..., k \tag{4.36}$$

Následně se provede prahování na základě pravděpodobnostního skóre.

Výsledkem by měly být oblasti QR-kódu.

Zhodnocení použitelnosti metody pro tuto práci

Metoda je jednoduchá, bohužel vyžaduje určení prahu prahování skóre pravděpodobnosti na konci algoritmu. Přesto pro hledání pouze jednoho QR-kódu, kdy se použije pouze určení dominantních úhlů a vzorkovací mřížky bez dělení snímku na bloky, je to velmi jednoduchý a účinný algoritmus.

5. Snímací scéna vhodná pro praktické použití

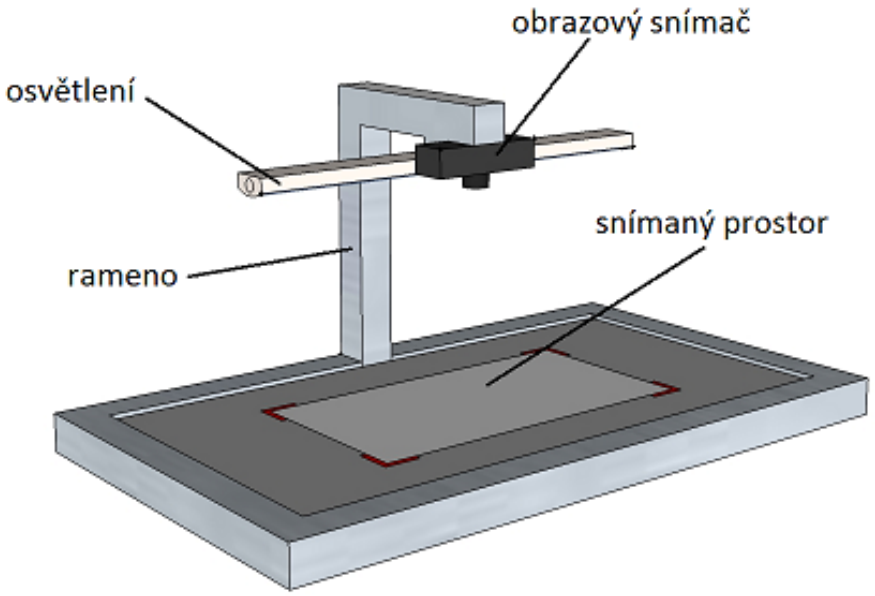
Pro nalezení kódu v obrázku musí být zásilka kvalitně nasnímána.

Samozřejmostí je zachování vzorkovacího teorému. Na nejmenší prvek kódu jsou zapotřebí minimálně dva vzorky. Takže rozlišení musí být dostatečné, aby připadaly minimálně dva pixely na nejmenší prvek kódu.

Pro dobré dekódování je také vhodné, aby měl snímek po celé ploše jednotné osvětlení, aby bylo možné snímek dobře vyprahovat.

Tato práce navrhuje řešení v podobě snímací stanice. Jedná se pouze o koncept bez komplexního konstrukčního návrhu.

Testovací snímky byly pořízeny pomocí externího osvětlení, běžného stativu nebo takzvaně z ruky. Přesto pro aplikaci snímání v zásilkových provozech se doporučuje sestavení snímacího zařízení, jehož možný koncept je načrtnut na obrázku 5.1.



Obrázek 5.1: Koncept přístroje na pořizování snímků pro čtení kódů.

Jedná se o podstavec s ramenem, na němž je umístěno světlo a obrazový snímač. Doporučuje se použití zářivky nebo pásu led diod s velkou svítivostí, které musí být rozmístěny tak, aby rovnoměrně osvětlovaly zásilku.

Pro snímání kódu vytištěných barvou citlivou na UV se místo bílého osvětlení namontuje osvětlení v příslušném ultrafialovém spektru a na snímací zařízení se přidají odpovídající filtry.

Rameno by mělo být teleskopické v případě, kdy se budou snímat zásilky od obálek až po větší balíky.

Obrazový snímač nebo celý přístroj se napojí na PC, které obsahuje vyhodnocovací software. Pro pohodlné spouštění snímání se doporučuje umístit na zařízení tlačítko spouště.

6. Pořízení testovací sady

Sada testovacích snímků byla pořizována pomocí:

- amatérské zrcadlovky Canon EOS 500D a setového objektivu Canon EF-S 18-55 mm f/3.5-5.6 IS II,
- mobilního telefonu Nokia 620.

Fotografování se provádělo z ruky. Každý testovací objekt byl fotografován kolmě seshora. Byla zde snaha zachytit každý objekt ve třech úrovních přiblížení kódu. V první úrovni byla snaha zachytit pokud možno celý objekt s kódem. V druhé úrovni byl zachycen kód a nějaké jeho okolí a ve třetí úrovni přiblížení byla snaha, aby kód vyplňoval celý snímek.

K tomu v každé z úrovní přiblížení byly pořízeny snímky objektu v pěti úhlech natočení. Začalo se snímat v správné orientaci objektu a poté se objekt vždy pootočil o $\pm 45°$ až po orientaci objektu otočenou o $\pm 180°$.

Tvorba testovacích objektů jednotlivých kódů bude popsána pro každý kód zvlášť v následujících kapitolách.

6.1. Testovací vzory IMB

V první řadě je důležité rozlišení snímků. Kód tvoří 65 pruhů. Na každý pruh jsou potřeba alespoň 2 pixely. Stejně tak na mezery mezi pruhy by měly připadat alespoň dva pixely. Mezer je 64. To je dohromady 258 pixelů. Z testovacího snímku 6.1 je zřejmé, že kód může tvořit 1/3 - 1/5 snímků, to je 778 - 1290 pixelů. V případě běžného snímku s poměry stran 3:2 a předpokladu, že kód je stejně orientován jako snímek, tak výška snímku bude 519 - 860 pixelů. Pro největší nutné rozlišení 1290×860 to odpovídá 1.11 MPx.

Jelikož v dnešní době mají fotoaparáty v mobilech 4 MPx a více, není rozlišení problém.

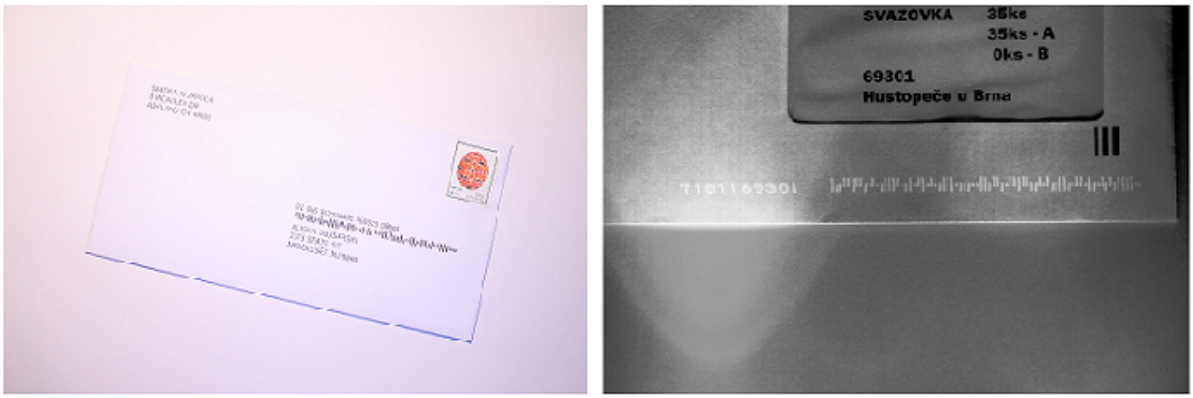
Jako testovací vzory byly použity obálky obdržené od České pošty. Tyto obálky mají čárový kód umístěný dole pod adresovým polem a vytištěný barvou citlivou na UV světlo. Jako zdroj světla sloužily LED diody emitující světlo o vlnové délce $395 - 400\,nm$.

Také byly použity obálky natištěné podle specifikací v dokumentaci [16]. Data na těchto obálkách jsou nesmysly, není účelem zpracovat data, ale přečíst kód. Jediné, co dává smysl, je směrovací číslo. Směrovací číslo z adresy na obálce je opravdu zakódované v čárovém kódu.

Podklady pro tisk byly vygenerovány pomocí Latexu. Kód a data v podobě vhodné pro čtení člověkem jsou generovány jako kód prostředí TikZ. Tento kód je generován funkcí v programu Matlab. Tato funkce byla napsána pro potřeby této práce. Do funkce vstupují data a výstupem je textový soubor, který po vložení do latexu vygeneruje čárový kód IMB, který kóduje vložená data a nad něj vloží tato data v podobě vhodné pro čtení člověkem.

Adresy na obálkách jsou náhodně vybrané ze souboru 500 adres určených pro testovací účely, které byly zdarma poskytnuty na stránkách Briana Dunninga [5].

Ukázky obálek jsou na obrázku 6.1.



a b

Obrázek 6.1: Testovací snímek s obálkou s potiskem a) podle dokumentace, b) UV barvou.

a b

Obrázek 6.2: Testovací snímek pro kód C128 a) standardní DL obálka, b) malá obálka.

6.2. Testovací vzory kódu C128 na doporučených zásilkách České pošty

Tyto vzory byly opět generovány pomocí skriptu v Matlabu jako seznam příkazů pro Latex. Po vytištění byly nalepeny na obálky s předtištěnými adresami.

Jako data adres byly použity náhodně zvolené adresy ze souboru adres [5]. Vstupní data pro kód jsou náhodná čísla, která nemají žádnou spojitost s realitou, natož nějaký význam.

Minimální rozlišení zde bude větší. Kód je dlouhý 4 cm a standardní DL obálka má délku 22 cm, na snímku tvoří 2/3 zabírané šířky. Celková délka je tedy $\pm 33\,cm$. Na kód České pošty pro doporučené zásilky je potřeba kód o délce 145 šířek nejužšího elementu. Podle Shannon-Kotelnikova teorému je tedy potřeba 290 pixelů na kód. Kód je 4/33 délky snímku, takže snímek by měl být ± 2400 pixelů dlouhý. Při poměru stran snímku 3:2 a orientace kódu stejné, jako je orientace snímku, je potřeba snímek o rozlišení $2400 \times 1600\,px$ to je $3.9\,MPx$. Pokud ale bude zabrána pouze oblast kódu, lze rozlišení snížit.

Ukázka testovacích snímků je na obrázku 6.2.

Obrázek 6.3: Testovací snímek kódu GS1-128 a) velký balík, b) malý balík.

6.3. Pořízení testovací sady snímků pro kódy GS1-128 na balíčcích

U tohoto kódu nebyl sestaven vlastní generátor, ale byl použit generátor dostupný na stránkách barcode.tec-it.com [32].

Data na štítcích opět postrádají význam. Štítky byly vytvořeny na základě podobnosti běžně používaným štítkům.

V tomto případě je otázka rozlišení poněkud problematická vzhledem k velké variabilitě délek kódů a jejich velikostí. Doporučuje se snímat pouze oblast štítku. Jelikož čárové kódy na balíčcích nebývají menší než kód České pošty na doporučených zásilkách, lze říci, že obecně by rozlišení snímku nemělo klesnout pod $3,9\,MPx$ pro stejné snímací podmínky jako je tomu u kódu C128.

Opět pro snímání pouze čárového kódu lze rozlišení snížit.

Ukázka testovacích snímků je na obrázku 6.3.

7. Popis tvorby algoritmů pro nalezení a čtení vybraných čárových kódů

Tato kapitola se v první řadě zabývá postupem při tvorbě algoritmů, které jsou schopné nalézt ve snímku příslušný čárový kód, provést jeho separaci od pozadí a následně ho i přečíst (dekódovat).

Vzhledem k rozdílnostem jednotlivých čárových kódů nelze navrhnout jeden univerzální algoritmus pro jejich nalezení ve snímku. Celkově se nedoporučuje v případě čtení kódů z celého snímku umísťovat čárové kódy různých druhů na jeden snímek, popřípadě na celý fotografovaný objekt.

U některých čárových kódů lze předpokládat apriorní znalosti o jejich umístění a vzhledu okolí. Tyto vlastnosti jsou často součástí celkového návrhu čárového kódu a bez jejich uvážení je nalezení kódu prakticky nemožné nebo velmi obtížné. Toto se projevuje zejména u IMB kódu, u kterého je a priori známo použití, vzhled objektu, na kterém je použit a jeho orientace vůči objektu.

Jsou zde podrobně rozebrány návrhy a nápady, které se objevily při vývoji algoritmů. Popsány jsou jak metody, které vedly k úspěšnému sestavení algoritmu, tak některé neúspěšné návrhy se zdůvodněním, proč je vhodnější použít jiné způsoby, i když se nápady mohly z počátku jevit jako dobré.

7.1. Návrh algoritmu pro nalezení a čtení kódů GS1-128 a C128

Jelikož je čárový kód České pošty C128 (2.2) založený na standardu pro čárový kód GS1-128 (2.1), přičemž C128 se liší formátem uložených dat, bude algoritmus popsán pouze na čárovém kódu GS1-128 a odlišnosti čtení kódu C128 budou zmíněny v průběhu.

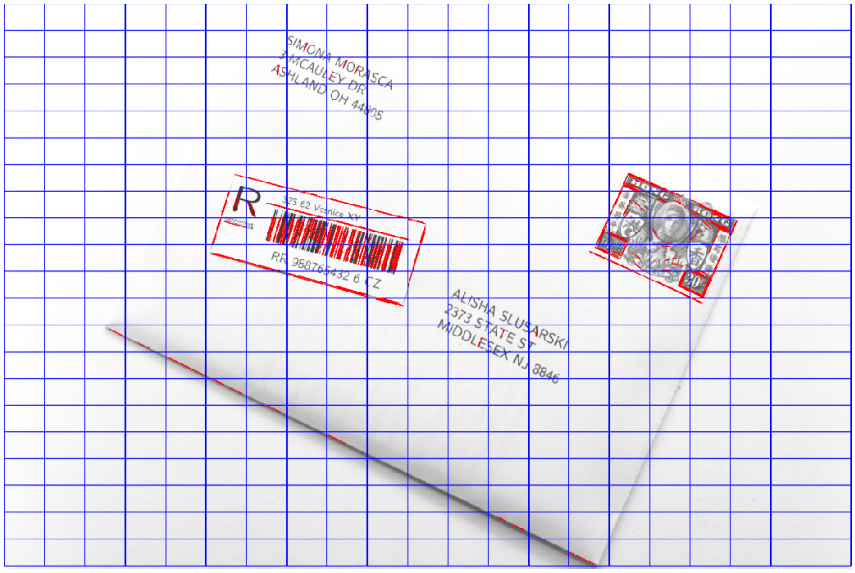
Standard GS1-128 umožňuje uživateli zvolit si vlastní metodu výpočtu kontrolního součtu. Pro zachování univerzálnosti čtecího algoritmu není implementován žádný výpočet kontrolního součtu a tím pádem ani jeho kontrola. Výstupem je tudíž jen řetězec nezpracovaných dat.

Situace je odlišná pouze u čárového kódu C128, kde je jen jediná možnost výpočtu kontrolního součtu, tudíž byl tento výpočet implementován a je součástí kontroly správnosti přečtení. Na výstupu je pak již předzpracovaný řetězec dat.

7.1.1. Předzpracování snímku s kódem GS1-128

Po načtení snímku se provede jeho převedení na odstíny šedé. Hodilo by se i prahování, ale vzhledem k možnosti rozdílného jasu v různých částech snímku je lepší provést prahování až po separaci oblasti kódu.

Dále je aplikován mediánový filtr, který má za úkol odstranit případný šum. Toto je vhodné provést vždy při práci se snímky, ale je třeba zvolit nastavení filtru tak, aby nedošlo k rozmazání snímků [2].



Obrázek 7.1: Ukázka rozdělení snímku kódu C128 na 21×21 buněk a zvýrazněné nalezené přímky.

7.1.2. Hrubé nalezení oblasti s kódem GS1-128

Čárový kód GS1-128 sestává z mnoha rovnoběžných pruhů, proto se nabízí nalezení oblasti kódu podle velkého množství rovnoběžných hran.

Za tímto účelem se snímek pomocí Cannyho detektoru hran (kapitola 4.1.4) převede na obraz hran.

Při prvních pokusech byla aplikována pravděpodobnostní Houghova transformace (kapitola 4.1.5) s nalezením přímek, která je velmi dobře implementována v používané OpenCV knihovně. Následně byl nalezen dominantní úhel natočení hran a podle něj snímek natáčen do správné orientace. Toto se bohužel ukázalo jako málo spolehlivé, a proto byla navržena modifikace této metody.

Nyní stále platí, že se snímek převede na obraz hran, ale poté se rozdělí na menší části (buňky), jejichž okraje tvoří mřížku. Počet buněk začíná na 21×21, což je nejvhodnější číslo pro nalezení kódu, který zabírá menší část snímku. V případě nenalezení celé oblasti kódu nebo špatného přečtení kódu se počet buněk iterativně zmenšuje až na počet 1×1, kdy je vyhodnocen celý snímek jako jedna buňka. Tato situace může nastat, pokud kód zabírá celý snímek.

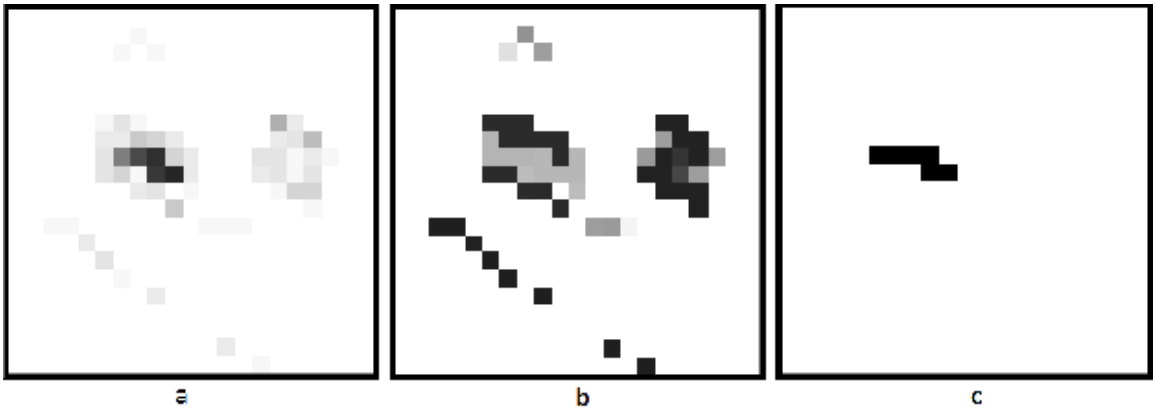
Ukázka rozdělení snímku a nalezení hran je na obrázku 7.1.

Jednotlivé buňky jsou vyhodnoceny samostatně a je jim přiřazena metrika. Jako metrika se používají tři parametry: dominantní úhel natočení hran v buňce, počet hran s dominantním úhlem a binární hodnota sousedství, která určuje, jestli má buňka souseda, který již byl vyhodnocen jako buňka obsahující část čárového kódu (dále jen sousedství). Se sousedstvím se pracuje později, a tak na začátku mají všechny buňky sousedství rovno 0.

Ilustrace procesu je na obrázku 7.2.

Po přiřazení metriky je nalezena buňka s největším počtem rovnoběžných hran, dále bude označována jako mateřská buňka a je jí přiřazeno sousedství 1.

Dále se vyhodnocuje okolí mateřské buňky. Začíná se s vyhodnocením buněk se vzdáleností 1 od mateřské buňky, to znamená 8 buněk, které mají společnou hranu nebo roh s mateřskou buňkou. V dalších krocích se berou buňky se vzdáleností 2, pak 3 atd.



Obrázek 7.2: Ukázky k textu o procesu hrubého hledání kódu GS1-128 ve snímku. a) Počet hran s dominantním natočením. b) Dominantní úhel natočení. c) Sousedství. (Hodnoty jsou upraveny pro lepší názornost.)

U vyhodnocované buňky se kontroluje, jestli má stejný dominantní úhel jako mateřská buňka (tolerance je $\pm 2°$) a jestli počet hran s dominantním úhlem není menší než 10 % hodnoty této metriky v mateřské buňce. Na konec se kontroluje, jestli buňka má souseda se sousedstvím 1 (platí osmiokolí). Pokud buňka vyhoví všem třem podmínkám, je jí přiřazeno sousedství 1.

Proces vyhodnocování končí po vyhodnocení všech buněk nebo pokud v prstenci buněk se stejnou vzdáleností od mateřské buňky není nalezena žádná vyhovující buňka.

Jako oblast s kódem se bere nejmenší oblast, která obsahuje všechny buňky, které mají sousedství 1, a je rozšířena o polovinu rozměru buňky. Rozšíření je vhodné pro případ, že by kód končil na hranici buňky a po výřezu by neměl žádné náběhové a doběhové zóny.

Na konec se provede výřez oblasti a její otočení o dominantní úhel mateřské buňky. Otočení se provádí bez ořezu na původní velikost. Výřez se nejdříve umístí do snímku o rozměrech úhlopříčky výřezu a až potom se otáčí.

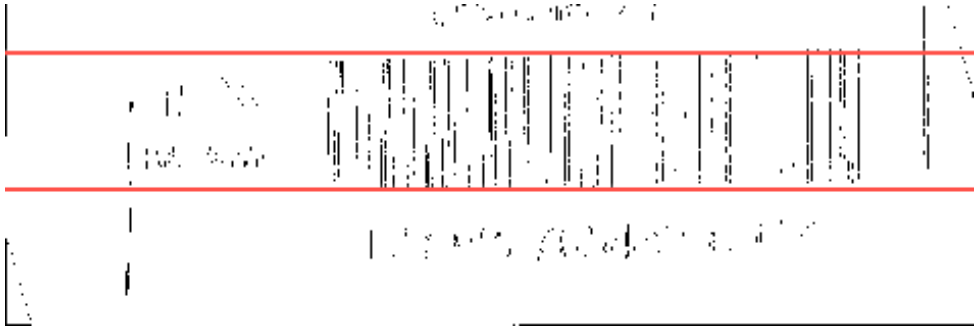
7.1.3. Přesné nalezení kódu GS1-128 v hrubém výřezu

Nyní je známá menší oblast, která obsahuje kód, ale ještě není vhodná pro čtení. Pro tento účel je potřebný přesnější výřez.

Pro určení horní a spodní hrany čárového kódu se výřez převede na obraz hran a tyto hrany jsou ještě dále filtrovány. Odstraní se všechny hrany, které nejsou kolmé (kód je správně orientován, tudíž má všechny hrany kolmě orientované). Body hran se vyhodnocují postupně a jako bod kolmé hrany je vyhodnocen bod, který má nad sebou, pod sebou nebo nad sebou i pod sebou dva hranové body.

Po filtraci hran se sečtou řádky obrazu hran. Teoreticky je největší hodnota ve vektoru sum řádků v místě čárového kódu, který má velké množství kolmých hran, proto se najde maximum ve vektoru. Od maxima se postupuje na obě strany a vyhodnocují se hodnoty ve vektoru, dokud tyto hodnoty neklesnou pod 10 % hodnoty maxima nebo se nedojde na konec/počátek vektoru. Místa ukončení se berou jako spodní a horní hranice čárového kódu. Znázornění nalezených hranic je na obrázku 7.3

Nyní lze provést výřez obrazu hran a přistoupit k hledání ohraničení kódu zleva a zprava. Pro tyto účely se obraz hran silně rozmaže Gaussovým filtrem. Účelem rozmazání



Obrázek 7.3: Znázorněná nalezená spodní a horní hranice čárového kódu C128. (Snímek je invertován.)

Obrázek 7.4: Ukázka hrubého rozmazání obrazu hran kódu C128. (Snímek má invertovány barvy a upravenou intenzitu jasu pro lepší názornost.)

je získat oblasti v odstínech šedi, ve kterých již nejsou patrné hrany a které lze vzhledem přirovnat k mlze (Obrázek 7.4).

Dále je postup obdobný jako u hledání spodní a horní hranice. Sečtou se sloupce a hledá se maximum ve vektoru sum sloupců. Od maxima se postupuje na obě strany vektorem, dokud vyhodnocované hodnoty vektoru neklesnou pod 10 % maxima nebo se nedosáhne konce/počátku vektoru. Tato místa jsou vyhodnocena jako levý a pravý okraj kódu. Rozšíření hranic pro získání náběhové a doběhové zóny není třeba provést. Rozmazání je takové, že se ve výřezu objeví i tyto bílé zóny.

Po těchto krocích jsou známy všechny čtyři hranice kódu a ten může být vyřezán z obrazu v odstínech šedi (obrázek 7.5a).

7.1.4. Prahování výřezu s kódem GS1-128

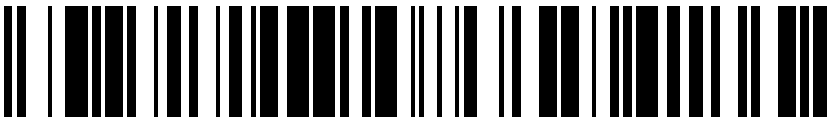
Volba správného prahování byla velmi obtížná. Běžné prahování nebylo možné z důvodu nemožnosti určit pevný práh ručně. Jako další připadalo v úvahu adaptivní prahování a Otsu prahování, ale ani tyto neobstály při pozdějších pokusech převést obraz na proud bitů.

Metodou experimentu byla jako nejvhodnější uznána metoda, která bude popsána nyní.

Nejprve bylo zapotřebí sjednocení jasu celé oblasti. Proto byl šedotónový výřez s kódem rozmazán Gaussovým filtrem tak, aby se ztratily veškeré detaily a zůstala pouze šedá plocha s jasovými přechody (obrázek 7.5b). Tato plocha byla brána jako mapa změn jasu a její inverze byla přičtena k výřezu s kódem. Výsledek je na obrázku 7.5c. Zvýšil se sice jas černých pruhů, ale nyní je jas pozadí konstantní a teoreticky se blíží hodnotě 255 (bílé barvě).

Dále by bylo možné provést klasické Otsu prahování, ale toto také nebylo nejvhodnější. Nejlépe se osvědčila metoda, kdy se sečnou sloupce šedotónového obrazu a provede se filtrace hodnot ve vektoru sum sloupců. Filtrace se provede tak, že všechny hodnoty

Obrázek 7.5: a) Výřez čárového kódu C128. b) Hrubé rozmazání (pozadí) výřezu. c) Výřez po odstranění pozadí.



Obrázek 7.6: Čárový kód C128 po prahování.

menší než 90 % hodnoty maxima ve vektoru jsou uznány jako černý pruh a zbylé jako bílý pruh. Výsledek je na obrázku 7.6.

7.1.5. Převod bílých a černých pruhů kódu GS1-128 na proud datových bitů

V této fázi je k dispozici vektor hodnot, které odpovídají barvám pruhů. Hodnota 0 odpovídá černým pruhům a všechny ostatní hodnoty odpovídají bílým pruhům.

Kód se skládá z bílých a černých pruhů, každý ve čtyřech různých šířkách. Nejtenčí pruh odpovídá jednomu bitu, nejtlustší čtyřem.

Jako první metoda převodu na proud datových bitů byla vyvinuta metoda, kdy se vypočte četnost šířek v kódu. Pro tyto účely byl vytvořen vektor, ve kterém tato četnost byla uložena na pozici odpovídající šířce pruhu. V takovém vektoru byla nalezena čtyři maxima a následně minima mezi nimi. Pozice těchto hodnot byly použity jako hraniční šířky pro přiřazení pruhů odpovídajících počtu bitů 1-4.

Toto fungovalo jen u vynikajících snímků s perfektní ostrostí a bez zkreslení. Proto bylo zapotřebí zavést modifikace.

První modifikací bylo, že se popsaný proces hledání hranic šířek provedl zvlášť pro bílé a zvlášť pro černé pruhy. Toto odstranilo problém s mírně rozostřenými snímky. Rozostření není nutně způsobeno špatným ostřením, ale vyplývá také z výskytu artefaktů na hranách, způsobených lomem světla v čočce. Výsledek však stále nebyl uspokojující.

Jako další modifikace bylo zavedeno hledání ne čtyř základních šířek, ale pouze jedné nejužší základní šířky (šířky jednoho bitu). Vše stále zvlášť pro bílé a zvlášť pro černé pruhy.

Základní šířka jednoho bitu se počítala z poloh čtyř maxim ve vektoru četnosti šířky pruhů, a to jako průměr z nejmenší šířky a rozdílů mezi sousedními šířkami. Vše znázorňuje vzorec 7.1.

$$s = \frac{s_1 + (s_2 - s_1) + (s_3 - s_2) + (s_4 - s_3)}{4} \tag{7.1}$$

kde s je průměr, s_1 je nejmenší šířka, s_2 je druhá nejmenší šířka, s_3 je třetí nejmenší šířka a s_4 je největší šířka.

Při převodu pruhů na proud datových bitů byly pruhy tříděné porovnáním s odpovídajícím násobkem určené nejmenší šířky s tolerancí poloviny určené nejmenší šířky.

Popsaný způsob převodu je používán v algoritmu. Jeho nevýhoda spočívá především v citlivosti na velké geometrické zkreslení kódu. Pokud by byl kód fotografován pod větším úhlem tak, že by se počet pixelů skládajících se na základní šíři pruhů značně měnil v celé délce kódu (perspektivní zkreslení), pak by byl kód pro algoritmus nečitelný.

Převod s využitím délky znaku a adaptací na geometrické zkreslení.

I když se v základu věci předpokládají nějaké podmínky pro čtení čárových kódů, aby byla zachována jejich dekódovatelnost, byl učiněn pokus vyřešit i problém geometrického zkreslení.

Tento problém s geometrickým zkreslením je řešitelný pomocí algoritmu, který se adaptuje na měnící se podmínky.

Tento algoritmus byl navrhnut tak, aby nepracoval se základní šířkou branou z celého kódu, nýbrž s jednotlivými znaky.

Každý znak je v GS1-128 kromě stop sekvence kódován třemi černými a třemi bílými pruhy a ty mají společně tloušťku 11 základních šířek (11 bitů). Z této znalosti vychází i převodní algoritmus, který si z vektoru hodnot odpovídajících barvám pruhů načte do pomocného vektoru hodnoty od počátku černého pruhu ke konci třetího bílého pruhu. Délka tohoto vektoru odpovídá jedenáctinásobku délky nejmenší základní šířky ve znaku. Po vydělení délky jedenácti je obdržena nejmenší základní šířka. Nyní se opět porovnávají šířky ve znaku s násobky nejmenší základní šířky a jsou jim přidělovány datové bity.

Převod na stop sekvenci se provádí obdobně a nebude zde již podrobně popisován.

U tohoto způsobu byly předběžně zaznamenány o něco horší výsledky, ale podrobnější vyhodnocení bude popsáno dále.

7.1.6. Převod proudu datových bitů na textový řetězec

Nyní je k dispozici čárový kód převedený na proud datových bitů. Ty se musí dekódovat pomocí dekódovací tabulky. Ta je uložena v algoritmu a pro její použití se znak v proudu bitů převede na dekadické číslo, to se nalezne v tabulce a jeho pozice je vrácena jako dekódovaná dekadická hodnota.

U algoritmu s korekcí geometrického zkreslení se tento převod provádí po každém nalezení znaku. Pokud nelze provést dekódování, nepovažuje se černý pruh za počátek znaku a pro hledání dalšího znaku se neskočí o tři černé pruhy, nýbrž jen o jeden.

Převést dekódované znaky na textový řetězec již není složité. Použije se rozhodovací strom, který postupně vyhodnocuje znaky, přepíná znakové sady, převádí dvoučíselná čísla na dva separované znaky v ASCI atd.

Podrobný popis převodu by byl více matoucí než užitečný, ale jeho prostota umožňuje udělat si vlastní představu o možnostech převodu.

7.2. Návrh algoritmu pro nalezení a čtení kódů IMB

Intelligent Mail Barcode, primárně navržený pro americký poštovní systém, se hojně používá i v České republice. Původně dokumentace předpokládá tisk černým inkoustem do oblasti adresy příjemce, ale nyní se používá i forma s potiskem barvou citlivou na jiné než viditelné spektrum v dolní části obálky. Pozice kódu na obálce je předem víceméně známá a kód je oddělitelný pomocí snímání v jiném spektru vlnových délek, proto nebyly v kódu umístěny výrazné obrazové prvky, které by umožnily jeho pohodlnou lokalizaci pomocí zpracování obrazu. Z tohoto důvodu jsou nutné apriorní znalosti, které umožní nalezení kódu ve snímku.

Především se jedná o znalost, že kód je paralelní k hranám obálky a ke směru textu. Dále se předpokládá, že kód je umístěn na obálce, kde nejsou žádné další podobné prvky ani příliš komplikované vzory. Přítomnost většího počtu paralelních hran v jiných prvcích na obálce může znemožnit nalezení IMB kódu.

7.2.1. Předzpracování snímku s kódem IMB

Je aplikován mediánový filtr, který je zvolen tak, aby odstranil jemný šum, ale tak, aby nerozmazal obrázek nebo nezničil významné detaily.

7.2.2. Nalezení hrubého úhlu natočení kódu IMB

Jelikož kód IMB neobsahuje výrazné paralelní hrany jako kódy skupiny GS1, je nutné hledat jiné specifikum než dominantní natočení hran, které umožní natočení snímku tak, že kód bude vodorovně orientován.

Knihovna OpenCV velmi dobře implementuje funkci pravděpodobnostní Houghovy transformace s hledáním přímek, kdy je možné zadat nejen minimální počet bodů hran, které tvoří přímku, ale i možné mezery v přímce.

Snímek byl nejdříve pomocí Cannyho filtru převeden na obraz hran a poté se v něm hledaly přímky pomocí pravděpodobnostní Houghovy transformace implementované v OpenCV.

Kód tvoří 65 pruhů, které po hranovém filtrování vytvoří 130 paralelních hran, proto byl parametr minimálního počtu bodů hran tvořících přímku nastaven na 120, přičemž snížení počtu je z důvodu možného nenalezení některých hran.

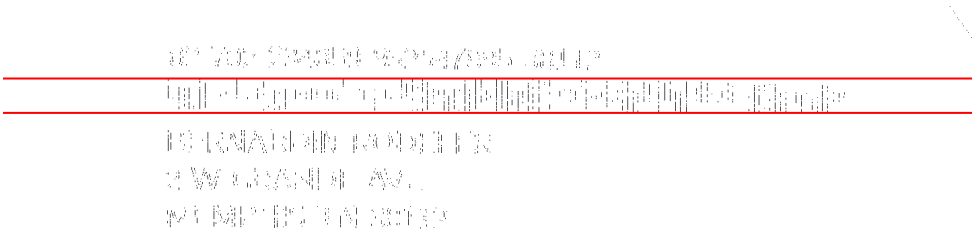
Druhý parametr, maximální mezery mezi body přímky, byl nastaven jako jedna stopadesátina většího rozměru snímku. Tato hodnota byla nalezena experimentální cestou.

Po provedení pravděpodobnostní Houghovy transformace je k dispozici seznam koncových bodů nalezených přímek. Z těchto bodů se určí úhel natočení přímek a následně vektor četnosti výskytu těchto úhlů. Předpokládá se, že nejčetnější úhel je hrubý úhel natočení kódu.

Ilustrace výsledku pravděpodobnostní Houghovy transformace je na obrázku 7.7

Se znalostí úhlu lze provést bezořezovou rotaci snímku.

Obrázek 7.7: Výsledek pravděpodobnostní Houghovy transformace s nalezením přímek při hledání hrubého úhlu natočení IMB kódu.



Obrázek 7.8: Nelezené horní a spodní hranice IMB kódu. (Snímek má invertované barvy.)

7.2.3. Nalezení horní a spodní hranice kódu IMB ve snímku

Nyní je k dispozice snímek IMB kódu, který je zhruba natočený do vodorovné pozice. Pro nalezení spodní a horní hranice kódu je potřeba provést Cannyho detekci hran a jejich následné filtrování.

Hrany se filtrují tak, aby zůstaly pouze vertikální hrany. Toho se dosáhne tím, že se odstraní všechny body hran, které nemají nad sebou, pod sebou nebo nad sebou i pod sebou dva jiné body hran.

Po filtraci hran se sečtou řádky snímku hran a provede se hledání maxima. Zde se předpokládá, že maximum se nachází v místě kódu.

Nyní se postupuje od maxima do obou směrů vektoru sum řádků. Horní a spodní hranice se umístí tam, kde hodnota ve vektoru klesne pod 10 % hodnoty maxima, nebo se nedosáhne konce/počátku vektoru. Ilustrace nalezených hranic je na obrázku 7.8.

Na konec se provede výřez pruhu snímku mezi hranicemi.

7.2.4. Nalezení levé a pravé hranice kódu

Výřez se převede na obraz hran a opět se vyfiltrují vertikální hrany.

Následně se obraz hran silně rozmaže Gaussovým filtrem. Účelem rozmazání je získat oblasti v odstínech šedi, ve kterých již nejsou patrné hrany a které lze vzhledem přirovnat k mlze.

Dále je postup obdobný jako u hledání spodní a horní hranice. Sečtou se sloupce a hledá se maximum ve vektoru sum sloupců. Od maxima se postupuje na obě strany vektorem, dokud vyhodnocované hodnoty vektoru neklesnou pod 10 % maxima nebo se nedosáhne konce/počátku vektoru. Tato místa jsou vyhodnoceny jako levý a pravý okraj kódu. Rozšíření hranic pro získání náběhové a doběhové zóny není třeba provádět. Rozmazání je takové, že se ve výřezu objeví i tyto bílé zóny.

Nakonec se provede výřez oblasti kódu.

7.2.5. Přesné nalezení úhlu natočení kódu

Pruhy v kódu IMB jsou dosti krátké a kód samotný je dlouhý, a tak i mírné pootočení celého kódu může vést k velkým deformacím. Proto je potřeba nalézt přesnější úhel natočení kódu.

Toto se provede převedením výřezu na obraz hran a aplikací pravděpodobnostní Houghovy transformace s hledáním přímek. Nyní se hledají přímky o délce jednoho obrazového bodu, jelikož se tento způsob osvědčil jako nejúčinnější. Výsledek je znázorněn na obrázku 7.9.

Opět se vypočte vektor četnosti natočení přímek a provede se rotace s ořezem o nejčetnější úhel.

Obrázek 7.9: Výsledek pravděpodobnostní Houghovy transformace s nalezením přímek při hledání přesného úhlu natočení IMB kódu.

7.2.6. Prahování výřezu s kódem IMB

Pro další čtení znaků kódu je potřeba provést prahování výřezu s kódem. Nejvhodnější je použití Otsu prahování nebo adaptivního prahování. V algoritmu je použito adaptivní prahování, jelikož u Otsu prahování se vyskytl problém při velké změně jasu na počátku a konci kódu. Jako velikost okna při adaptivním prahování se použije počet řádků výřezu s kódem.

Prahování se provádí inverzí, takže tam, kde je pruh, bude hodnota 255 a tam, kde je pozadí (mezera), hodnota 0.

7.2.7. Filtrování výřezu s IMB kódem pro odstranění nežádoucích prvků

Kód IMB má výhodu, že se skládá z pruhů stejné šířky a mezer, které také mají každá stejnou šířku. Této vlastnosti lze využít pro odstranění nežádoucích prvků v prahovaném snímku.

Provede se sumace sloupců výřezu s kódem. Vektor sum se převede na binární vektor, tam, kde je hodnota větší než 0, se uloží 1. Dále se pracuje s tímto binárním vektorem.

Filtrace podle šířky pruhů

Vypočte se vektor četnosti šířek pruhů (délek spojitých úseků jedniček). Nalezne se nejčastější šířka a z vektoru se vymažou úseky jedniček, které mají délku jinou než z rozsahu: nejčastější šířka ± jedna třetina této šířky.

Filtrace podle šířky mezer

Účelem této filtrace je odstranit artefakty z počátku výřezu s kódem.

Vypočte se vektor četnosti šířek mezer mezi pruhy (délek spojitých úseků nul) a určí se nejčetnější šířka.

Poté se postupuje binárním vektorem, a pokud je nalezena mezera (úsek nul) mimo rozsah: nejčastější šířka ± jedna polovina této šířky, vynulují se hodnoty od počátku binárního vektoru až po tuto nevyhovující mezeru.

Pokud je nalezeno více jak 10 mezer z rozsahu po sobě, jedná se již s největší pravděpodobností o kód a proces je přerušen.

O oblast za kódem není třeba se starat, jelikož algoritmus čte jen prvních 65 pruhů.

Filtrace pomocí Fourierovy transformace

Provede se diskrétní Fourierova transformace binárního vektoru a odstraní se frekvence s amplitudou nižší než 5 % maximální amplitudy. Následně se provede zpětná diskrétní Fourierova transformace.

Aplikace filtrovaného binárního vektoru na prahovaný výřez

Nyní se aplikuje filtrovaný binární vektor na výřez s kódem. Tato aplikace je jednoduchá, ve výřezu se ponechají neporušené pouze sloupce, které odpovídají jedničkovým hodnotám v binárním vektoru. Zbylé sloupce se vynulují.

Výsledek je na obrázku 7.10.

Obrázek 7.10: Výsledný výřez s kódem IMB po filtraci nežádoucích artefaktů. (Snímek má invertované barvy.)

7.2.8. Převod výřezu s kódem na vektor hodnot pruhů

Nejprve je potřeba nalézt pozice horizontálních středů všech pruhů. Toho se dosáhne hledáním středů mezi hranami pruhů ve vektoru sum sloupců.

Dále je zapotřebí určit tři základní výšky pruhů. V místech horizontálních středů pruhů se sečtou sloupce a vypočte se vektor četnosti délek. Jelikož se jedná o binární, prahovaný obraz, suma bude odpovídat počtu bílých bodů ve sloupci.

Ve vektoru sum se najdou tři maxima. Maxima se hledají tímto způsobem: po nalezení jednoho maxima se postupuje vektorem na obě strany od maxima a nulují se hodnoty, dokud ty klesají nebo se nenarazí na nulu, popřípadě konec/počátek vektoru. Poté se opět hledá maximum.

Následně se převádí pruhy na hodnoty 0, 1, 2 a 3, které kódují typ pruhu.

Nejprve se určí hranice mezi krátkým, středním a dlouhým pruhem. Tyto hranice se určí ze tří základních výšek pruhů a výpočet se provede podle rovnice 7.2 a 7.3:

$$\text{spodní hranice} = \text{nejmenší výška} + \frac{\text{střední výška} - \text{nejmenší výška}}{2} \tag{7.2}$$

$$\text{horní hranice} = \text{střední výška} + \frac{\text{největší výška} - \text{střední výška}}{2} \tag{7.3}$$

Typ pruhu se určí ze vzhledu sloupce v místě horizontálního středu pruhu. Vypočtou se dvě hodnoty. První je suma od středu sloupce na horu a druhá je suma od středu sloupce dolů.

Pokud suma těchto hodnot je menší než spodní hranice, uloží se hodnota 0 - krátký pruh. Pokud je suma vetší než horní hranice, přidělí se hodnota 3 - dlouhý pruh. Pokud je hodnota mezi hranicemi, provede se další rozhodování.

Pokud je suma od středu sloupce nahoru větší než suma od středu dolů, jedná se o stoupající pruh a je přidělena hodnota 1, v opačném případě se jedná o klesající pruh a je přidělana hodnota 2.

Původní návrh počítal se separací pruhů podle jejich levé a pravé hranice. Následně se počítal průměr, později medián z horní a spodní poloviny vzniklého obdélníku. Tento způsob nebyl spolehlivý z důvodu náchylnosti na chyby při malém natočení kódu, které algoritmus nedorovnal v předchozích krocích.

7.2.9. Převod pruhů IMB kódu na proud bitů

Ze zpracování snímku je k dispozici vektor obsahující zakódovanou posloupnost pruhů a jejich typ. Nyní je potřeba vygenerovat posloupnost bitů. Jelikož rozložení bitů v kódu je dáno tabulkou v dokumentaci [16], je potřeba provést převod pomocí této tabulky. Algoritmus má v sobě uložené dvě tabulky, jednu pro klesající a druhou pro stoupající pruhy.

Postupně se prochází vektor zakódovaných pruhů. Vyskytne-li se pruh kódovaný 1 nebo 2, použije se odpovídající tabulka, ve které se nalezne pořadí bitu odpovídající danému pruhu a na toto místo se uloží jednička. Pokud se vyskytne pruh kódovaný trojkou, musí se provést uložení jedničky pomocí obou tabulek.

Výsledkem je proud 130 bitů.

7.2.10. Nalezení a oprava chyb v proudu bitů kódu IMB

Dekódovací proces vychází z procesu kódování (kapiptola 2.3), který se obrátí.

Nejdříve se proud bitů rozdělí na deset znaků po třinácti bitech a v nich se spočte počet jedničkových bitů.

Následně se určí počet znaků, které mají nesprávný počet jedničkových bitů. Pokud se vyskytne alespoň jeden chybný znak, provede se rotace kódu o 90° a opět se provede překlad na proud bitů a vyhodnotí se počet chybných znaků.

Dále se pracuje s orientací kódu, která má méně chybných znaků.

Pokud je nyní počet chybných znaků větší než nula, je zahájen opravný algoritmus. Ten nejdříve zkontroluje, kolik chybných znaků musí opravit. Pokud je to více jak 4, kód se vyhodnotí jako neřešitelný. Kontrolní součet je uložen v kódu pomocí počtu jedniček

ve znacích, a tudíž se mění v průběhu opravného algoritmu, který pracuje na základě změn bitů v chybných slovech. Při opravě příliš velkého počtu chyb již dochází k hádání a výsledek nemusí být správný. Při opravě maximálně 4 chyb je pravděpodobnost shody kontrolního součtu při chybném čtení dostatečně malá.

Samotný opravný algoritmus pak testuje všechny kombinace, které lze vytvořit změnou jednoho bitu v chybných znacích tak, aby měly korektní počet jedniček. Například pokud chybný znak obsahuje 10 jedniček, je nejbližší správný počet jedniček 11. Potom s největší pravděpodobností je jedna jednička zaměněná na nulu. Potom opravný algoritmus postupně testuje všechny variace, kdy zamění jednu nulu na jedničku.

Opravný algoritmus je ukončen, pokud najde kombinaci oprav, které dají proud bitů, jenž má shodný jak vypočtený, tak uložený kontrolní součet.

Může se stát, že je v jednom znaku více chyb. Takovéto chyby algoritmus neumí opravit. Naštěstí kód obsahuje opatření, která mohou shluk chyb v tištěném kódu rozdělit do více znaků.

7.2.11. Dekódování znaků na slova v kódu IMB

V tomto kroku již budeme předpokládat, že máme proud bitů rozdělen na znaky se správným počtem jedniček.

Jak již bylo zmíněno u popisu opravného algoritmu, počítá se počet jedníček ve znacích. Z tohoto počtu lze určit 10 bitů uloženého kontrolního součtu.

Následně je nutné provést bitovou negaci znaků, které jsou spárované s bitem kontrolního součtu rovným jedné. (Pro bližší pochopení problému se doporučuje studium kapitoly 2.3.2, která popisuje způsob kódování dat do IMB kódu a je zde také podrobně popsán způsob uložení kontrolního součtu do kódu.)

Po negaci příslušných znaků lze provést dekódování znaků na slova. Slova jsou získána převodem ze znaků pomocí tabulek. Tabulky jsou dvě a jsou generovány pomocí funkce uvedené v dokumentaci [16].

Algoritmus u překladu pracuje se znaky a slovy v podobě dekadických čísel. Proto je nejdříve zařazen převod znaků v podobě bitů na dekadické číslo.

Po dekódování lze určit poslední, jedenáctý bit uloženého kontrolního součtu. Pokud je první slovo (slovo A) větší než 659, uloží se jedenáctý bit kontrolního součtu roven jedné a zároveň se hodnota prvního slova zmenší o 659. V opačném případě se uloží hodnota nula a slovo A se nemění.

7.2.12. Převod slov na data a výpočet kontrolního součtu z dat v kódu IMB

Nyní je potřeba provést se slovy pár operací sčítání a násobení. Jedná se o opačný algoritmus než při kódování (kapitola 2.3.2) a nebude zde již podrobně popisován.

Jelikož je výsledek obrovské číslo, které se nevejde do žádného běžného datového typu používaného v jazyce C++, je potřeba výsledek uložit do jiného formátu. Tímto formátem je pole třinácti *unsigned char* prvků, které jako celek tvoří binární podobu výsledného čísla.

Vzhledem k způsobu uložení čísla, bylo zapotřebí operace sčítání a násobení provést pomocí speciálně pro tyto účely navržených funkcí. Jedná se o funkce, které pracují s již

zmíněným polem *unsigned char* prvků, jako s číslem v 2^8 soustavě. Například pokud třetí prvek pole je roven 124, druhý 5 a první 46, jedná se o číslo $124 \cdot (2^8)^2 + 5 \cdot (2^8)^1 + 46 \cdot (2^8)^0 = 8\,127\,790$.

Po převodu slov na číslo uložené v poli třinácti *unsigned char* prvků je potřeba vypočíst kontrolní součet. Zde opět specifikace [16] přináší hotovou funkci.

Nakonec se provede porovnání vypočteného kontrolního součtu s kontrolním součtem uloženým v kódu. Pokud se součty neshodují, je indikováno špatné dekódování.

7.2.13. Převod dat dekódovaných z IMB kódu na čitelný formát

Posledním problémem je prezentace získaných dat. Číslo uložené v poli s třinácti *unsigned char* prvky po vytištění zdánlivě nedává smysl. Proto je potřeba číslo převést do soustavy, která je lidskému myšlení bližší.

Pro tyto účely byla vytvořena funkce, která převede pole, které se chová jako číslo v 2^8 soustavě, na pole, které se chová jako číslo v 10 000 soustavě. Takovéto pole potom obsahuje v každém prvku čtyři pozice výsledného čísla v dekadické podobě.

8. Vyhodnocení algoritmů na testovacích snímcích

Algoritmy pro čtení čárových kódů byly zkoušeny na sadách testovacích snímků a výsledku budou představeny v následujících kapitolách.

Snímky byly pořizovány z ruky pomocí zrcadlovky Canon EOS 500D a mobilního telefonu Nokia 620.

8.1. Vyhodnocení algoritmů čtení kódu České pošty C128

Jako testovací objekty bylo zvoleno osm obálek s nalepeným čárovým kódem C128. Obálky byly fotografovány ve třech stupních přiblížení a v pěti polohách pro každý stupeň přiblížení. Snímky byly pořizovány kolmo seshora.

Při pokusech na první sadě pořízené zrcadlovkou Canon EOS 500D byl algoritmus neúspěšný. Po prozkoumání důvodů bylo zjištěno, že testovací sada obsahuje velké množství rozostřených snímků. Vyskytuje se zde problém s omezeným ostřícím rozsahem objektivu, který je zvláště patrný při fotografování zblízka. Proto byla vytvořena druhá sada snímků, které byly pořízeny s větší pečlivostí a s důrazem na dobré zaostření.

Nová sada pro testy spolehlivosti algoritmů je již vyhovující a obsahuje 120 snímků.

Sada byla testována na dvou typech algoritmů, které byly popsány v kapitole 7.1, a to algoritmus čtení podle statistických vlastností šířek pruhů z celého kódu a algoritmus čtení po znaku s využitím známosti délky znaku. Úspěšnost algoritmů je znázorněna v tabulce 8.1.

Výpočet úspěšnosti byl proveden tak, že se úspěšně přečtenému snímku přiřadila hodnota 1 a neúspěšně přečtenému snímku hodnota 0. Po otestování všech snímků se jejich hodnoty sečetly a tato hodnota se vydělila počtem snímků. Výsledek je prezentován v procentuální formě.

Typ snímače	**Typ algoritmu**	**Správně vyhodnoceno**
Canon EOS 500D	Využití statistik šířek pruhů	96 %
Canon EOS 500D	Využití známosti délky znaku	33 %
Typ snímače	**Typ sady**	**Správně vyhodnoceno**
Nokia 620	Kompletní sada 120 snímků	61 %
Nokia 620	Sada vybraných 80 snímků	88 %

Tabulka 8.1: Úspěšnost algoritmů čtení kódu České pošty C128.

Velkým problémem se ukázalo prostorové zkreslení kódu, které je způsobeno vadou objektivu, takzvaného rybího oka. Toto se hlavně projevuje na malých ohniskových vzdálenostech u levnějších objektivů a u méně kvalitních objektivů v mobilních telefonech. Řešením je použití dražších objektivů s pevnou ohniskovou vzdáleností.

K dalšímu zkreslení dochází v důsledku perspektivy, zde je jediným řešením co nejkolmější snímání kódu.

I když by se mohlo zdát, že si s problémem geometrického a perspektivního zkreslení lépe poradí druhý algoritmus, který se adaptuje pro každý znak, není tomu tak. Tento algoritmus zklamal veškerá očekávání a měl úspěšnost pouze 33 %.

První algoritmus, který pracuje se základní šířkou, která je určena ze statistiky šířek z celého kódu, je velice úspěšný a správně přečetl 96 % testovaných snímků.

Další testy byly prováděny na testovací sadě pořízené pomocí mobilního telefonu Nokia 620. Zde vyvstal problém s nižším rozlišením snímků. Algoritmus nebyl schopen přečíst kódy na snímcích, na kterých byla celá obálka i s částí okolí. Proto jsou v tabulce 8.1 uvedeny dvě úspěšnosti. První je pro kompletní sadu 120 snímků a je rovná 61%. Druhá úspěšnost je pro zmenšenou sadu 80 snímků, která neobsahuje snímky, na kterých je čárový kód v příliš malém rozlišení. V tomto případě dosáhl algoritmus úspěšnosti v 88 % případů.

Snímky z mobilního telefonu byly vyhodnocovány pouze prvním algoritmem. Jelikož algoritmus čtení s využitím délky znaku dosáhl v prvních testech mnohem horších výsledků, byl z dalšího testování vynechán.

Koncové závěry z testu algoritmů

Čárový kód C128 nebyl navržen jako kód pro čtení pomocí kamery (snímkování), proto je zapotřebí zachovat nějaké předpoklady pro umožnění čtení ze snímku.

Pokud se kód snímá v dobrém osvětlení a přímo seshora a přitom samotný kód není nijak deformován a je v odpovídajícím rozlišení, lze ho úspěšně přečíst.

Je vhodné použít objektiv s malou vadou nebo používat ohniskové vzdálenosti, kde objektiv projevuje menší náklonnost k vadám. Dále se doporučuje dbát na správné zaostření snímku. Za dobrých podmínek je možné přečíst i jemně rozostřený kód, ale úspěšnost čtení se rapidně sníží.

Dále je zde požadavek na rozlišení snímků, které by mělo být pokud možno co největší. Původní předpoklad, že stačí dodržet vzorkovací teorém a vystačit si se dvěma pixely na nejmenší šířku, selhal. Je vhodné, aby na nejmenší základní šířku připadalo alespoň $5-10$ pixelů.

Problém rozlišení a zkreslení je patrný především u mobilních telefonů, které mají často fotoaparáty s menším rozlišením, špatnými šumovými vlastnostmi a jsou vybaveny velmi levnou a nekvalitní optikou.

8.2. Vyhodnocení algoritmu čtení kódu GS1-128

U tohoto algoritmu nebyl vyhodnocován způsob čtení pomocí známosti délky znaku, jelikož v testech čtení kódu C128 dopadl tento způsob oproti druhému způsobu velice špatně. Nemělo proto smysl opět testovat oba způsoby.

Pro pořízení testovacích snímků bylo použito pět štítků, které byly snímány ve třech stupních přiblížení a v pěti polohách pro každý stupeň. Vzniklo celkem 75 snímků. Bližší popis snímaných objektů je v kapitole 6.

Výpočet úspěšnosti probíhal obdobně jako u kódu C128 s tím rozdílem, že na jednom snímku se mohlo nacházet více čárových kódů. V tomto případě při správném přečtení jen části kódů byla snímku přiřazena poměrově odpovídající hodnota. Například pokud byly kódy tři a byly přečteny všechny, byla snímku přiřazena hodnota 1, pokud ale byly přečteny pouze dva kódy, byla snímku přiřazena hodnota 0.66.

Snímky z testovací sady pořízené pomocí zrcadlovky Canon EOS 500D byly správně přečteny v 94 % případů a snímky pořízené mobilním telefonem Nokia 620 v 91 % případů.

Typ snímače	Správně vyhodnoceno
Canon EOS 500D	94 %
Nokia 620	91 %

Tabulka 8.2: Úspěšnost algoritmu čtení kódu GS1-128.

Jelikož testované čárové kódy GS1-128 jsou větších rozměrů než testované kódy C128, nebyl zde problém s rozlišením snímků pořízených mobilním telefonem.

Závěry z testu algoritmů

Platí zde stejné závěry jako pro snímky s kódem C128 popsané v předešlé kapitole 8.1, a proto zde nebudou opakovány.

Je třeba zmínit, že vzhledem k tomu, že kód GS1-128 může dosahovat značných délek, je zde třeba klást ještě větší důraz na zmenšení vlivu vady objektivu, perspektivy nebo na deformaci kódu samotného.

8.3. Vyhodnocení algoritmu čtení kódu IMB

Jak již bylo zmiňováno, tento kód neobsahuje významné prvky, které by umožňovaly jeho pohodlné nalezení, proto je potřeba dodržet požadavky na vzhled potištěného objektu a tisk kódu samotný. Podrobný popis fotografovaných objektů je v kapitole 6.

V konečném důsledku byly pořízeny tři sady po 75 snímků fotografovaných v kolmém směru. Každá sada obsahuje snímky pěti obálek v pěti úhlech natočení pro tři různé stupně přiblížení.

Při pořizování sady testovacích snímků mobilním telefonem Nokia 620 panovaly specifické světelné podmínky. Do místnosti, ve které probíhalo fotografování, svítilo slunce skrze větve stromu a to přímo na místo, kde byly umístěny fotografované testovací obálky. Tyto světelné podmínky byly využity pro pořízení testovací sady, která nemá rovnoměrné osvětlení. Výsledky jsou pro tuto sadu horší než pro později pořízenou sadu s rovnoměrným osvětlením, ale i přesto jsou uspokojivé. Konkrétně pro sadu nerovnoměrně osvětlených snímků se jedná o úspěšnost 84 % a pro sadu s rovnoměrným osvětlením o úspěšnost 99 %.

Snímky pořízené pomocí zrcadlovky Canon EOS 500D byly správně vyhodnoceny v 97 % případů.

Typ snímače	Typ sady	Správně vyhodnoceno
Canon EOS 500D	s rovnoměrným osvětlením	97 %
Nokia 620	s rovnoměrným osvětlením	99 %
Nokia 620	s nerovnoměrným osvětlením	84 %

Tabulka 8.3: Úspěšnost algoritmu čtení kódu IMB.

Závěry z testu algoritmů

Algoritmus dává velmi uspokojivé výsledky pro všechny testovací sady. Oproti kódům skupiny GS1 zde neplatí tak silný požadavek na ostrost, ale přesto by snímky měly být co nejostřejší pro co nejspolehlivější čtení.

Jelikož dokumentace umožňuje i tenčí provedení pruhů kódu IMB, než tomu bylo u testovacích snímků, je potřeba dbát na dostatečné rozlišení. Z pokusů lze říci, že dostatečné rozlišení, při kterém nedochází k problémům při vyhodnocování prvků obrazu, je 10 obrazových bodů na šířku pruhu. Samozřejmě čím více, tím lépe. Naopak pokud rozlišení klesne pod 5 obrazových bodů na šířku pruhu, je čtení velmi obtížné.

9. Závěr

Byly prozkoumány tři typy čárových kódů používaných na poštovních zásilkách. Konkrétně se jedná o tři nejpoužívanější kódy. Kód Intelligent Mail Barcode 2.3 používaný na psaních, kód GS1-128 2.1 pro identifikaci a popis vlastností balíků a jeho varianta C128 používaná Českou poštou pro identifikaci doporučených psaní 2.2 a univerzální QR-kód 2.4, kterým se kódují stejné informace jako kódem GS1-128 nebo se v něm uchovává odkaz na web se službami spojenými se zasíláním balíků. QR-kód samozřejmě může uchovávat jakékoliv dodatečné informace.

Ve zmíněných podkapitolách s popisem kódů byl rozebrán způsob předzpracování dat do podoby proudu bitů a připojení kontrolních a opravných bitů, dále způsob převodu na samotný kód do podoby pruhů nebo čtverečků.

Samostatná kapitola byla věnována postupu při výpočtu korekce chyby u QR-kódu (kapitola 3). Tato korekce vychází z Reed-Solomonovy korekce chyby a pracuje s polynomy a aritmetikou Galoisova tělesa. Postup výpočtu je popsán stručným a pokud možno přehledným způsobem tak, aby na jeho základě bylo možné přistoupit k tvorbě algoritmu, který řeší korekci chyby při dekódování QR-kódu.

Dále byly prozkoumány současné metody detekce kódů v obraze. Nejprve byly probrány obecné nástroje, které se vyskytují v algoritmech detekce kódů, přičemž některé algoritmy využívají speciální matematické aparáty, které byly popsány přímo při algoritmu. Bohužel se k velmi komerčnímu kódu IMB nepodařilo nalézt žádné dostupné články, ale díky vstřícnosti pracovníků třídírny dopisů České pošty v Brně byl získán přístup k třídicímu stroji, který kód IMB využívá. Jeho popis je dostupný v podkapitole 4.2.

Nejvíce článků je věnováno kódům EAN-13 používaným na spotřebním zboží. Tyto algoritmy jsou použitelné i pro kód GS1-128, jelikož jeho vzhled je velmi podobný. Podrobné zhodnocení jednotlivých algoritmů se nachází u každého algoritmu zvlášť. Obecně lze říci, že většina algoritmů vyžaduje přibližnou znalost velikostí kódů ve snímku. Je zde patrný rychlý vývoj, kdy u starších algoritmů jsou použity metody, které se dnes dají nahradit jinými a efektivnějšími nástroji.

Jako nejvíce vhodná se jeví metoda, kdy se obraz převede na hrany nebo skelety a aplikuje se Houghova transformace pro nalezení dominantního úhlu natočení hran. Po rotaci kódu do vodorovné polohy se provede segmentace na základě lokálních vlastností. Kódy mají paralelní hrany, což je jedna z vlastností, podle které se dá snímek lehce segmentovat.

Většina algoritmů je údajně použitelná i pro 2D maticové kódy, především QR-kód, kdy se algoritmus rozšíří o výpočet ve dvou dimenzích. Příkladem je metoda pro detekci QR-kódu popsaná v kapitole 4.3.6, která je zřejmě nejlepší a zároveň poměrně jednoduchá a účinná. Existuje zde také nepřeberné množství algoritmů, které se snaží detekovat QR-kód pomocí dnes tak módních neuronových sítí. Tyto metody, zdá se, jsou velmi náročné na vývoj a neposkytují tak závratně lepší výsledky než jiné metody.

Jako nejvhodnější kód pro zasílané psaní se jeví kód IMB, který je jednoduchý a přitom obsahuje množství ochranných prvků, díky nimž je schopný detekovat a opravit chyby. Co tento kód neumí, je uchování většího množství dat. Toto zase umožňuje kód GS1-128, který je nejvhodnější pro použití na balíčcích. Tento kód byl vytvořen pro potřeby popisu balíků a dokumentace [9] , popisuje množství standardizovaných formátů uložení dat (jako příklad lze uvést datum spotřeby nebo váhu balíku). Kód GS1-128 bude také velmi snadno detekovatelný, protože obsahuje velký počet paralelních hran. Česká pošta

používá kód GS1-128 s vlastním formátem uložených dat a výpočtem kontrolního součtu pro popis a identifikaci doporučených zásilek (označení kódu je C128). Nevýhodou kódu GS1-128 je poměrně slabé zabezpečení kontrolním součtem pro detekci chyby a nemožnost automatické opravy.

V kapitole 5 bylo popsáno zařízení, které by mohlo být použito pro čtení kódů, například ve skladech. Jedná se pouze o koncept a nebylo sestaveno.

Pro účely testování a vývoje algoritmů byly vytvořeny sady snímků. Jejich tvorba a tvorba testovacích objektů jsou popsány v kapitole 6, ale také v kapitole 8 přímo se věnující vyhodnocení navržených algoritmů na těchto testovacích sadách. Problémem bylo pouze snímání obálek potištěných barvou citlivou na UV světlo. Jako zdroj ultrafialového světla byly použity LED diody. Po osvětlení těmito diodami je kód zřetelný pro člověka, ale pro běžný fotoaparát bohužel ne, jelikož čipy v těchto zařízeních jsou vybaveny UV filtrem a primárně snímají viditelné spektrum záření. Pro snímání těchto kódů by bylo zapotřebí použít černobílý snímač, který UV-filtr nemá, nebo specializované zařízení.

Popis algoritmů schopných najít a dekódovat kódy GS1-128, C128 a IMB je v kapitole 7. Je zde snaha vše popsat takovým způsobem, aby bylo jasné, jak se k jednotlivým řešením dospělo a proč bylo použito zrovna toto řešení a ne jiné. Kódy skupiny GS1 mají velké množství paralelních hran, proto na základě této vlastnosti nebylo problémem jejich nalezení. Oproti tomu bylo více problémů s detekcí IMB kódu, který nemá výrazné obrazové prvky, které by umožnily jeho pohodlné nalezení ve snímku. Proto pro čtení tohoto kódu pomocí zpracování obrazu musí být kladeny požadavky i na snímaný objekt s kódem. Bližší popis těchto požadavků je v kapitole 7.2.

Jak již bylo zmíněno, vyhodnocení algoritmů je popsáno v kapitole 8. Procentuální úspěšnost jednotlivých algoritmů na příslušných testovacích sadách je přehledně zobrazena v tabulkách 8.1, 8.2 a 8.3. Obecně lze konstatovat, že algoritmy mají úspěšnost nad 90 % u kvalitních snímků s rovnoměrným osvětlením a dobrým rozlišením. Jako dobré rozlišení lze brát případy, kdy alespoň deset obrazových bodů připadá na nejmenší šířku prvků kódu. Pro méně kvalitní snímky, zvláště ty pořízené mobilním telefonem Nokia 620 a ty pořízené za špatných světelných podmínek, se úspěšnost stále pohybuje nad 80 %. Proto lze konstatovat, že navržené algoritmy, při dodržení požadavků popsaných v kapitole 8, jsou vhodné pro nasazení při čtení čárových kódů.

Tato práce by se mohla dále zabývat zdokonalením algoritmů detekce a čtení čárových kódů. Důležitým a přínosným rozšířením této práce by bylo přidání kompenzace prostorového a geometrického zkreslení způsobeného především perspektivou. Toto by umožnilo čtení kódů i z fotografií pořízených pod úhlem, a ne jen z těch focených kolmo.

Další možností navázání na tuto práci je praktické využití poznatků o Reed-Solomonově korekci chyby u QR-kódu při vývoji algoritmů, popřípadě by bylo vhodné provést podrobnější rozbor matematiky v pozadí celého problému, zde především aritmetiky Galoisova tělesa.

Literatura

[1] BAŞARAN, E.; Özgür ULUÇAY; ERTÜRK, S.: READING BARCODES USING DIGITAL CAMERAS THROUGH IMAGE PROCESSING. *Proceedings of 5th International Symposium on Intelligent Manufacturing Systems*, ročník 8, Květen 2006: s. 835–843.

[2] Beneš, B.; Sochor, J.; Felkel, P.; aj.: *Moderní počítačová grafika*. Computer press, 1998, ISBN 80-7226-049-9.

[3] Bodnár, P.; Nyúl, L. G.: Improving Barcode Detection with Combination of Simple Detectors. *Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on Naples*, Listopad 2012: s. 300–306.

[4] Chai, D.; Hock, F.: Locating and Decoding EAN-13 Barcodes from Images Captured by Digital Cameras. *Information, Communications and Signal Processing, 2005 Fifth International Conference on Bangkok*, 2005: s. 1595–1599.

[5] Dunning, B.: Free Sample Data for Testing. Říjen 2013.
URL http://www.briandunning.com/sample-data/

[6] Eby, C.: Error Correction Coding. Říjen 2014.
URL http://www.thonky.com/qr-code-tutorial/error-correction-coding/

[7] Felzenszwalb, P. F.; Huttenlocher, D. P.: Distance Transforms of Sampled Functions. *THEORY OF COMPUTING*, ročník 8, Září 2009: s. 415–428.

[8] Gremlins, B.: Image Moments. Prosinec 2014.
URL https://breadboardgremlins.wordpress.com/image-moments/

[9] Version 14, GS1 General Specifications. Leden 2014.

[10] Horák, K.: Matematická morfologie. 2013.

[11] Horák, K.; Kalová, I.; Petyovský, P.; aj.: Počítačové vidění. Duben 2008.

[12] Hu, T.; Yang, T.: Application of Computational Verb Image Processing to 1-D Barcode Localization. *Anti-Counterfeiting, Security and Identification (ASID), 2013 IEEE International Conference on Shanghai*, Říjen 2013: s. 1–4.

[13] Jirsík, V.: Umělé neuronové sítě. 2012.

[14] Jirsík, V.: Vícevrstvá nauronová síť s algoritmem učení backpropagation. 2012.

[15] Kalová, I.: Segmentace a detekce geometrických primitiv. 2012.

[16] Milligan: INTELLIGENT MAIL® BARCODE (4-STATE CUSTOMER BARCODE). Listopad 2006.

[17] Richter, M.: Fourierova transformace ve 2D. Leden 2011.

[18] Sonka, M.; Hlavac, V.; Boyle, R.: *Image Processing, Analysis, and Machine Vision*. Thomson, třetí vydání, Březen 2007, ISBN 049508252X.

[19] Szentandrási, I.; Herout, A.; Dubská, M.: Fast Detection and Recognition of QR codes in High-Resolution Images. *SCCG '12 Proceedings of the 28th Spring Conference on Computer Graphics*, Březen 2013: s. 129–136.

[20] Wikipedia: Forney algorithm — Wikipedia, The Free Encyclopedia. 2014, [Online přístup 23-04-2015].
URL http://en.wikipedia.org/wiki/Forney_algorithm

[21] Wikipedia: Fourierova transformace — Wikipedia, The Free Encyclopedia. Září 2014.
URL http://cs.wikipedia.org/wiki/Fourierova_transformace

[22] Wikipedia: Reed–Solomon error correction — Wikipedia, The Free Encyclopedia. 2015, [Online přístup 23-04-2015].
URL http://en.wikipedia.org/wiki/Reed%E2%80%93Solomon_error_correction#Syndrome_decoding

[23] Wikiversity: Reed–Solomon codes for coders — Wikiversity. 2014.
URL http://en.wikipedia.org/wiki/Forney_algorithm

[24] Yang, T.: Applications of Computational Verbs to Digital Image Processing. *International Journal of Computational Cognition*, ročník 3, Září 2005: str. 32, ISSN 1542-8060.

[25] Labeled Inventory. Září 2014.
URL http://www.amazon.com/gp/help/customer/display.html?nodeId=200243200

[26] Intelligent Mail Barcode Decoder. Říjen 2014.
URL http://bobcodes.weebly.com/imb.html

[27] EASYLABEL® 5 Etiket Design Software. 2012.
URL http://www.intermec.info/software/easylabel-etiket-design-software/easylabel-5-etiket-design-software.html

[28] Pokyny České pošty pro označování doporučených zásilek čárovými kódy; Hromadní podavatelé. Duben 2014.

[29] Information technology — Automatic identification and data capture techniques —+ QR Code 2005 bar code symbology specification. 2006.

[30] Kanji Charakters. Říjen 2014.
URL http://www.japan-zone.com/new/kanji.shtml

[31] WZÓR-ADRESOWANIA-PE24-P24-P48. Září 2014.
URL http://www.poczta-polska.pl/

[32] Online Barcode Generator.
URL http://barcode.tec-it.com/

[33] Zamberletti, A.; Gallo, I.; Albertini, S.: Robust Angle Invariant 1D Barcode Detection. *Pattern Recognition (ACPR), 2013 2nd IAPR Asian Conference on Naha*, Listopad 2013: s. 160–164.

A. Datový nosič DVD

Datový nosič DVD obsahuje:

1. Elektronická verze této práce ve formátu PDF
2. Zdrojové kódy algoritmu detekce a kódování GS1-128 - jako projekt pro Microsoft Visual Studio 2013
3. Zdrojové kódy algoritmu detekce a kódování C128 - jako projekt pro Microsoft Visual Studio 2013
4. Zdrojové kódy algoritmu detekce a kódování IMB - jako projekt pro Microsoft Visual Studio 2013
5. Testovací sada snímků Canon EOS 500D - C128
6. Testovací sada snímků Nokia 620 - C128 - kompletní sada
7. Testovací sada snímků Nokia 620 - C128 - sada vybraných snímků
8. Testovací sada snímků Canon EOS 500D - GS1-128
9. Testovací sada snímků Nokia 620 - GS1-128
10. Testovací sada snímků Canon EOS 500D - IMB
11. Testovací sada snímků Nokia 620 - IMB - s rovnoměrným osvětlením
12. Testovací sada snímků Nokia 620 - IMB - s nerovnoměrným osvětlením
13. Podrobné výsledky testů - ve formátu XLSX
14. Testovací čárové kódy C128
15. Testovací obálky s IMB kódem - potisk
16. Testovací obálky C6 bez kódu - potisk
17. Testovací obálky DL bez kódu - potisk
18. Testovací štítky s kódem GS1-128
19. Generátor kódu C128 pro Latex jako funkce pro Matlab
20. Generátor kódu IMB pro Latex jako funkce pro Matlab