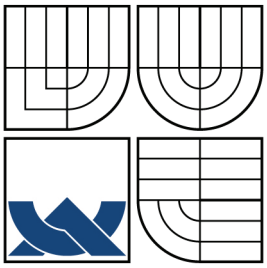
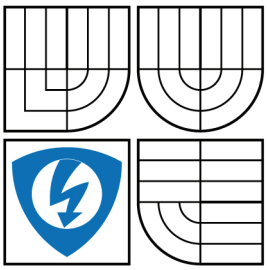


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

SYSTÉM PRO ROZPOZNÁVÁNÍ 2D ČÁROVÝCH KÓDŮ

2D BARCODE RECOGNITION SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN SEDLÁŘ
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR PETYOVSKÝ
SUPERVISOR

BRNO 2010

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student: Martin Sedlář
Ročník: 3

ID: 106764
Akademický rok: 2009/10

NÁZEV TÉMATU:

Systém pro rozpoznávání 2D čárových kódů

POKYNY PRO VYPRACOVÁNÍ:

1. Prostudujte problematiku čárových kódů.
2. Seznamte se s postupy a metodami detekce čárových kódů používanými v průmyslu.
3. Na základě nastudovaných znalostí zvolte vhodný systém uspořádání 2D čárových kódů, případně navrhněte vlastní.
4. Navrhněte vhodné uspořádání optické scény pro snímání 2D čárových kódů pomocí CCD kamery a vytvořte množinu testovacích snímků.
5. Navrhněte a realizujte algoritmy pro optickou detekci a rozpoznání 2D čárových kódů.
6. Zhodnoťte dosažené výsledky, určete přesnost klasifikace 2D čárových kódů na základě vyhodnocení testovacích snímků.

DOPORUČENÁ LITERATURA:

[1] Šonka, M.; Hlaváč, V.: Počítačové vidění, Grada, Praha 1992, ISBN 80-85424-67-3
[2] Žára, J.; Beneš, B.; Felkel, P.: Moderní počítačová grafika, Computer press, 1998, ISBN 80-7226-049-9
[3] Hlaváč, V.; Sedláček, M.: Zpracování signálů a obrazů, skriptum ČVUT 2001

Termín zadání: 8.2.2010 ***Termín odevzdání:*** 31.5.2010

Vedoucí práce: Ing. Petr Petyovský
Konzultanti bakalářské práce:

prof. Ing. Pavel Jura, CSc.
předseda oborové rady

UPOZORNĚNÍ:

ANOTACE

Tato bakalářská práce se zabývá čtením 2D čárových kódů. Na začátku je uveden stručný přehled nejpoužívanějších čárových kódů. Dále se hlavně práce zabývá QR kódy. Vysvětlena je jejich struktura, složení a pravidla pro jejich tvorbu a čtení. Nastíněna je problematika uspořádání scény při jejich snímání. Popsán je algoritmus pro čtení QR kódů, který je psaný v jazyce C++ a využívá knihovnu OpenCV.

ANOTATION

This bachelor's thesis deals with 2D barcode reading. At the beginning a brief overview of the most used barcodes is shown. The work mainly deals with QR codes. Their structure, composition and rules for encoding and decoding are explained. Problems with arrangement of photographic scene are solved too. C++ algorithm for QR code reading is described with usage of the OpenCV library.

KLÍČOVÁ SLOVA

QR kód, 2D kód, čárový kód, zpracování obrazu, knihovna OpenCV, Cannyho hranový detektor, Houghova transformace, uspořádání scény

KEYWORDS

QR code, 2D code, barcode, image processing, OpenCV library, Canny edge detector, Hough transform, scene arrangement

Bibliografická citace

SEDLÁŘ, M. Systém pro rozpoznávání 2D čárových kódů. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 65 s. Vedoucí bakalářské práce Ing. Petr Petyovský.

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma SYSTÉM PRO ROZPOZNÁVÁNÍ 2D ČÁROVÝCH KÓDŮ jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské 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 § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne: **27. května 2010** ……
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Petrovi Petyovskému za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: **27. května 2010** ……
podpis autora

OBSAH

SEZNAM POUŽITÝCH OBRÁZKŮ

SEZNAM TABULEK

1. ÚVOD

Tato práce se zabývá problematikou čtení QR kódů. Na začátku je uveden stručný přehled všech čárových 1D i 2D kódů, které se ve velkém rozsahu používají v průmyslu, logistice, ale částečně i v jiných aplikacích, se kterými se lze setkat v každodenním životě. Významně urychlují kontrolu produktů, při jejich pohybu, výrobě, či montáži.

Z těchto kódů byly vybrány právě QR kódy. Z užívaných kódů mají, například oproti EAN kódu, propracovanější korekci chyb a vzhledem k jejich velikosti je možné do nich uložit větší množství dat. V této práci je uvedeno jejich složení, důležité prvky v nich obsažené, způsoby kódování dat, atd. Jejich struktura, vlastnosti a kódování dat jsou tak složité, že je tato problematika rozdělena na několik podkapitol. Dále je také uveden příklad převodu dat do QR kódu krok po kroku.

V další kapitole je popsán postup zhotovení databáze snímků reálného QR kódu, vytištěného na matném a lesklém papíru. Byly použity různé typy osvětlení a uspořádání scény tak, aby byly pořízeny co nejkvalitnější snímky.

Další kapitola popisuje algoritmus zpracování snímku QR kódu, s využitím programovacího jazyku C++ a knihovny OpenCV. Jeho vstupem je zmíněný snímek, výstupem jsou dekódovaná data. V závěru této práce jsou uvedeny dosažené výsledky.

2. TEORETICKÝ ÚVOD

V této kapitole je krátce zmíněna historie čárových kódů a oblast jejich použití. V první části je uveden přehled nejpoužívanějších čárových kódů. V druhé části je detailně popsán QR kód, jeho struktura, složení, vlastnosti a je zde také uveden příklad zakódování dat do QR kódu.

2.1 ČÁROVÉ KÓDY

Kapitola obecně popisuje vznik a použití čárových kódů. Uvádí dělení čárových kódů na 1D a 2D kódy a nejznámější zástupce z těchto skupin.

2.1.1 Jak vznikl čárový kód

První patent na čárový kód byl podán již v roce 1949 dvěma muži, Bernardem Silverem a Normanem Josephem Woodlandem. Silver se snažil vyhovět požadavku prezidenta obchodních firem na vývoj systému k automatickému čtení údaje o produktu při kontrole. Ke komerčnímu využití došlo až roku 1966, a to ve tvaru soustředných kružnic, tzv. „býčí oko.“ Tento tvar se neosvědčil, a proto roku 1970 vznikl klasický čárový kód složený z čar. Roku 1974 vznikla jeho první čtečka a produkt s čárovým kódem se objevil v obchodech. [16]

2.1.2 Co je to čárový kód a kde se používá

Čárový kód je v dnešní době velice efektivní způsob, jak rychle a bezchybně načítat data do počítače či jiného systému k dalšímu zpracování, aniž bychom použili klávesnici. Podle zdroje [7] dochází při ručním zadávání k chybě průměrně při každém třístém zadání, zatímco u čárových kódů se počet chyb snižuje až na jednu miliontinu. Většina kódů obsahuje i zabezpečení, díky kterému lze zjistit, zda přečtená data jsou korektní nebo ne. Navíc se do některých čárových kódů vnáší i korekce chyb, která sice zabere určité místo, jinak určené pro data, ale dokáže zpětně zrekonstruovat a opravit poškozený kód. QR kód dokáže například opravit až 30%

poškozené matice. Do čárového kódu se může ukládat výrobní číslo, váha, cena, název výrobce, datum výroby, jméno zodpovědné osoby, skladové informace, atd.

Při tvorbě čárového kódu se využívá kontrastních barev jako je černá a bílá, ať už v podobě černých čar a bílých mezer nebo černých a bílých čtverců. Kontrastu se využívá při čtení čtečkami, založenými na principu odrazu infračerveného nebo spíše laserového paprsku, ale také při snímání kamerou s následnou detekcí hran mezi černou a bílou barvou pomocí počítačového zpracování obrazu. Čárový kód může být také vypálen laserem přímo na výrobek. [5], [7], [10].

2.1.3 1D čárové kódy

1D (jednorozměrné) čárové kódy = skládají se z několika vodorovných černých čar o různých tloušťkách oddělených bílými mezerami. Čtení tedy probíhá jen v jedné souřadnici. Jsou v Evropě prozatím nejrozšířenějšími čárovými kódy, ale v budoucnu by je mohly nahradit 2D maticové kódy.

EAN

„Zkratka EAN znamená European Article Number. Nejčastější EAN kód a pravděpodobně nejčastější čárový kód vůbec je EAN-13, který byl definován standardizační organizací GS1. Kódy EAN-13 jsou používány po celém světě k označování jednotlivých druhů zboží. Pozměněná podoba tohoto kódu například umí uchovávat ISBN kódy knižních publikací nebo ISSN kódy časopisů a jiných periodik. Z kódu EAN-13 lze zjistit zemi původu výrobce nebo způsob užití daného zboží. Méně jsou používány kódy EAN-8, které jsou vyhrazeny a používány pro menší položky, na které je problém umístit 13místný kód, jako jsou třeba cukrovinky“. Citováno z [13].

U kódu EAN-13 platí, že první dvě nebo tři číslice jsou systémové. Obvykle určují zemi, kde je zaregistrován výrobce, nebo konverzi do ISBN nebo ISSN. Další čtyři nebo pět číslic obsahují kód výrobce. Následující pětice určuje kód výrobku. Poslední je kontrolní číslice, která se většinou dopočítá funkcí modulo 10. Každá číslice je kódována dvěma čárami a dvěma mezerami. [13]

Obr. 1 - Ukázka kódu EAN-13 [13]

UCC/EAN 128

„UCC/EAN128 je čárový kód využívaný pro označování obchodních a logistických jednotek. Umožňuje zakódovat pomocí standardizovaných aplikačních identifikátorů (AI) mnoho podstatných informací o daném výrobku, jako jsou např. číslo dodávky, datum výroby, datum balení, minimální trvanlivost, hmotnost, délka, šířka, plocha, objem, komu má být zboží zasláno atd. Každá z informací má svůj vlastní AI, který jednoznačně určuje o jaký typ údaje se jedná. Pro vlastní kódování znaků se využívá Code 128.“ Citováno z [8].

Obr. 2 - Ukázka kódu UCC/EAN 128 [8]

CODE 39

„Kód Code 39 je používaný v automobilovém průmyslu, ve zdravotnictví i v dalších odvětvích průmyslu a obchodu. Je schopen kódovat číslice 0 až 9, písmena A až Z a dalších sedm speciálních znaků, přičemž každý znak je reprezentován pěti čárami a čtyřmi mezerami.“ Citováno z [10].

Obr. 3 - Ukázka kódu Code 39 [10]

Kód 2 z 5 (2/5)

Je možné zakódovat pouze numerická data. Kód obsahuje znak Start a Stop. Každá číslice se skládá z pěti čar a pěti mezer, z toho 3 čáry jsou slabé a 2 čáry jsou silné (trojnásobek slabé čáry). Dobré ke čtení ve ztížených podmínkách a nekvalitně vytištěných kódů. Dosahují však příliš velké délky, která je navíc proměnná. [13]

Obr. 4 - Ukázka kódu 2 z 5 [14]

2.1.4 2D čárové kódy

2D (dvourozměrné) čárové kódy = skládají se většinou z matice černých a bílých čtverců (modulů), tím jsou data uložena v obou souřadnicích, jak horizontálních, tak vertikálních. Začíná se rozšiřovat jejich užití, díky implementované korekci chyb u většiny z nich a větší datové kapacitě. S ní však roste i náročnost kódu na jeho zpracování.

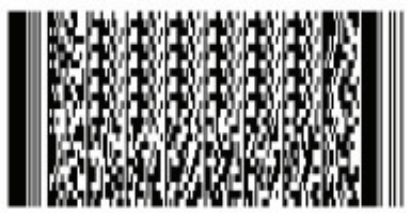
PDF 417

„PDF 417 je dvourozměrný (2D) kód s velmi vysokou informační kapacitou a schopností detekce a oprav chyb (při porušení kódu). Označení PDF 417 (Portable Data File) vychází ze struktury kódu: každé kódové slovo se sestává ze 4 čar a 4 mezer o šířce minimálně jednoho a maximálně šesti modulů. Celkem je však modulů ve slově vždy přesně 17. Na rozdíl od 1D čárových kódů, které obvykle slouží jako klíč k vyhledání údajů v nějaké databázi externího systému, si PDF 417 nese všechny údaje s sebou a stává se tak nezávislý na vnějším systému.

Do PDF 417 lze zakódovat nejenom běžný text, ale i grafiku nebo speciální programovací instrukce. Velikost datového souboru může přitom být až 1,1 kB. Při

generování symbolu lze zvolit úroveň korekce chyb, čímž lze zabezpečit čitelnost i při částečném poškození kódu.

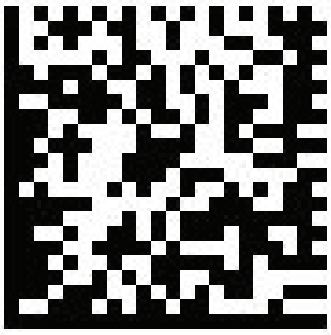
Příkladem použití PDF 417 mohou být nejrůznější identifikační karty, řidičské průkazy (v některých státech USA), kód lze využít i pro zakódování diagnózy pacientů apod. Kódy PDF 417 jsou rovněž využívány v systému EAN/UCC v kombinaci s EAN 13, UPC A, UCC/EAN 128 a GS1 Databar kódy jako tzv. složené (kompozitní) kódy.“ Citováno z [9].



Obr. 5 - Ukázka kódu PDF 417 [9]

DATAMATRIX

Je 2D kód, který se skládá z matice čtverců. Referenčním znakem je levý a dolní okraj, který je tvořen souvislými čárami. Může být ve velikostech od 8x8 do 144x144 bodů. Maximální objem dat je 2335 pro alfanumerické znaky a 3116 pro numerické znaky. Datamatrix používá Reed-Solomonovu korekci chyb. [7][13]



Obr. 6 - Ukázka kódu DATAMATRIX [13]

QR KÓD

Kód je velice podobný Datamatrix kódu, ale obsahuje jiné referenční obrazce. Nejsou to jen dvě čáry, ale jsou to tři prvky, z nichž každý se skládá ze tří soustředných čtverců, což usnadňuje nalezení kódu ve scéně a jeho správnou rotaci při zpracování. Objem zakódovaných dat může být 4296 alfanumerických znaků, 7089 numerických znaků, ale i Byte (binární data) a Kanji znaky. Použita je Reed-Solomonova korekce chyb.

Obr. 7 - Ukázka QR kódu

2.2 QR KÓD

2.2.1 Úvod

QR kódy jsou popsány v normě ISO/IEC 18004, ze které bylo převážně čerpáno v celé této kapitole a je uvedena jako zdroj [5]. V následujících podkapitolách je uvedeno, z čeho se QR kódy skládají, k čemu slouží jejich jednotlivé prvky, jaká data mohou být do kódů uložena a kolik jakého typu jich může být, informace o způsobu převodu vlastních dat do tvaru pro vložení do kódu a dodatečných informačních datech kódu. V Tab. 1 je uvedeno, kolik znaků lze uložit do QR kódu. Ten je vzhledem k jeho velikosti schopen nést velké množství dat. Implementovány jsou korekce chyb, takže v nejvíce zabezpečeném případě lze obnovit až 30% poškozených dat. Tyto kódy jsou charakteristické svými soustřednými čtverci ve třech ze čtyř rohů, které urychlují lokalizování kódu ve scéně. Kód obsahuje i další prvky, které zkvalitňují a zrychlují čtení kódu.

Typ dat	Počet znaků
Numerická data	7089
Alfanumerická data	4296
Byte	2953
Kanji	1817

Tab. 1 - Maximální počet znaků v QR kódu

2.2.2 Struktura QR kódu

QR kód se skládá z matice černých a bílých bodů (modulů). Přičemž černý modul představuje logickou jedničku a bílý logickou nulu.

Záchytným bodem je tzv. Finder pattern. Nazýván také jako kotvící obrazec. Využívá se k rychlému lokalizování levého horního rohu kódu. Skládá se ze tří soustředných čtverců jak je ukázáno na Obr. 8 i s jeho přesně definovanými rozměry.

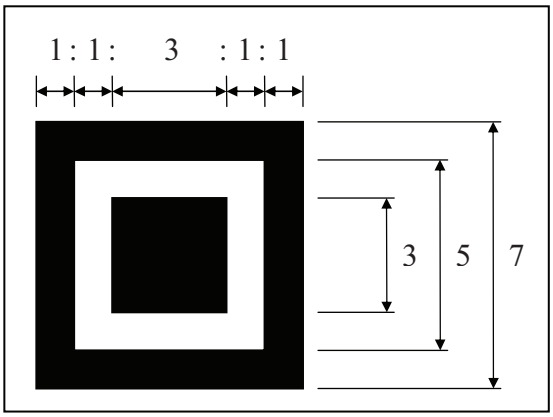
Od velikosti kódu verze 2 přibývá ještě další tzv. Alignment pattern a se zvyšující se verzí přibývá jejich počet. Jeho struktura je zobrazena na Obr. 9 a

příklad umístění je na Obr. 11. Tento obrazec slouží k synchronizaci souřadnic na pořízeném snímku kódu a souřadnic matice černých a bílých bodů (=modulů) tohoto kódu v reálném prostředí. Pokud by byl kód vytištěn na mírně nerovném (zvlněném) povrchu, pomohou tyto obrazce při zpracování a narovnání zdeformovaného obrazu.

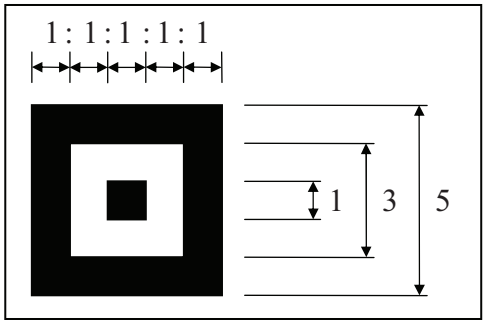
Dalším důležitým prvkem QR kódu jsou Timing patterns (= zaměřovací značky). Skládají se z jednoho řádku (sloupce) střídajících se modulů černé a bílé barvy, přičemž vždy se začíná a končí černým modulem. Určují tak hustotu souřadnicové sítě QR symbolu.

Separátor slouží k oddělení dat a referenčních symbolů. Jeho šířka je 1 bod (modul). Quiet zone (= prázdný okraj kódu) je tvořen oblastí o šířce 4 bodů kolem celého QR symbolu a umožňuje čtecím zařízením rychleji zaměřit kód ve scéně, aniž by splýval s okolím.

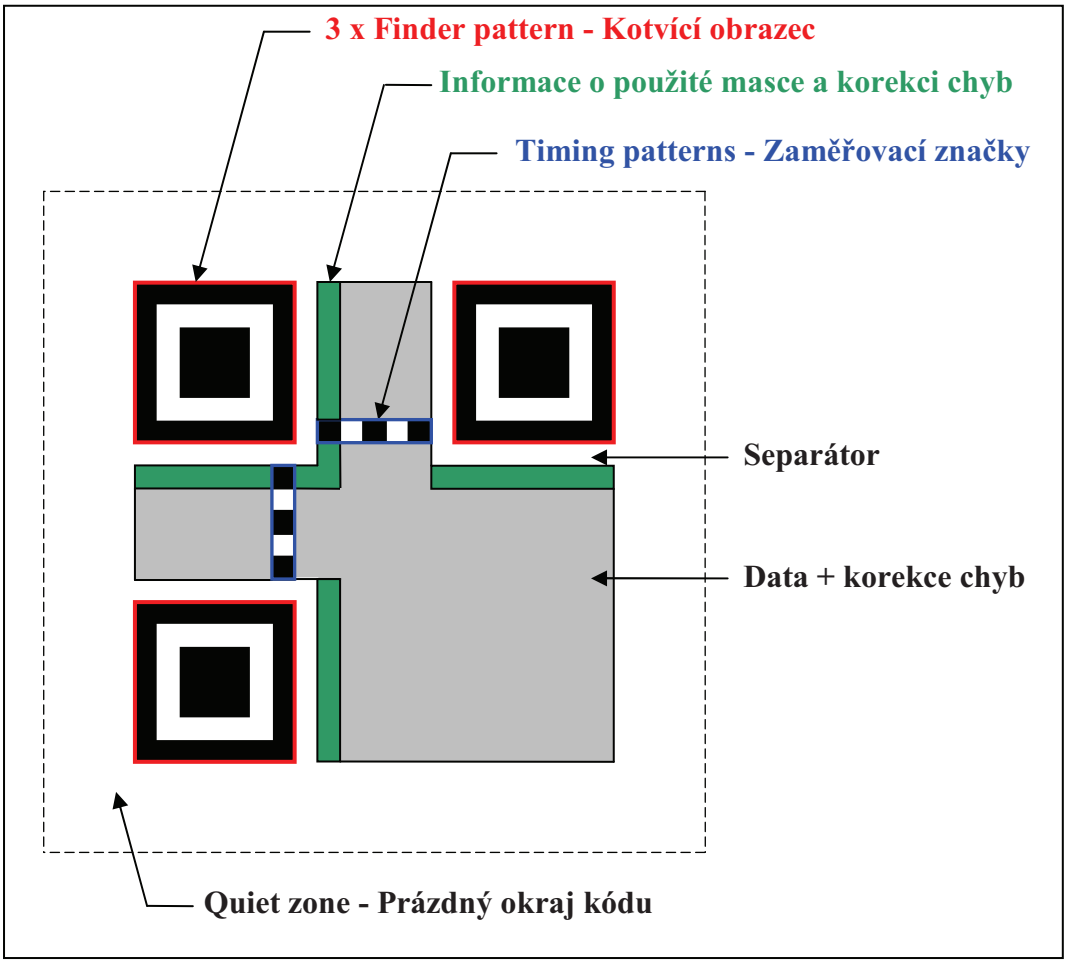
Zbývající oblast QR kódu je určena pro vlastní data, data korekce chyb, informaci o použité masce a typu korekce chyb. Celá struktura kódu je zobrazena na Obr. 10. U kódu verze 2 a vyšší přibývá navíc zarovnávací symbol, jehož počet a umístění se mění s použitou verzí (velikostí) kódu podle přesných pravidel, a také přibývá údaj o verzi kódu, kdy u verze 2 představuje datový prostor 7 bitů a zvyšuje se s verzí kódu dle pravidel. Tento údaj o verzi kódu obsahuje i korekci chyb. Kód verze 2 je na Obr. 11.



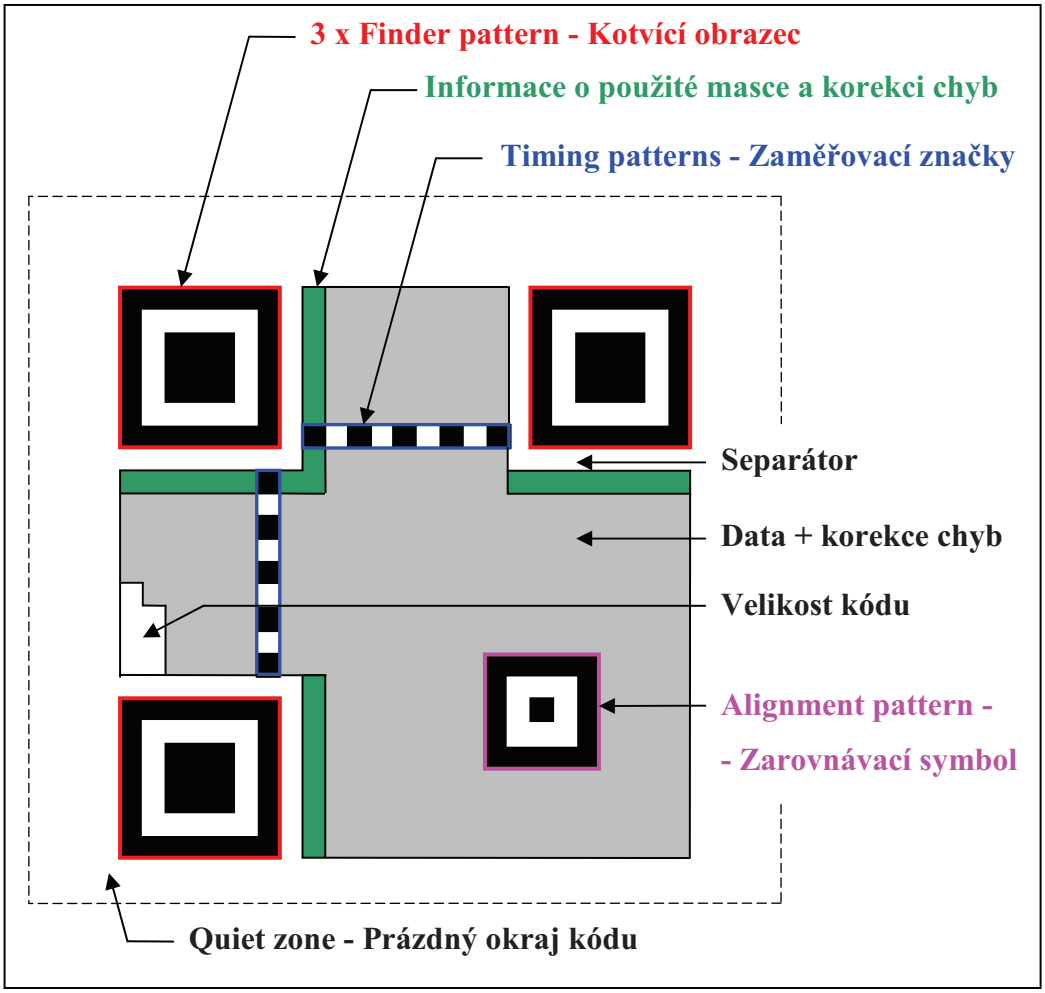
Obr. 8 - Struktura Finder pattern



Obr. 9 - Struktura Alignment pattern



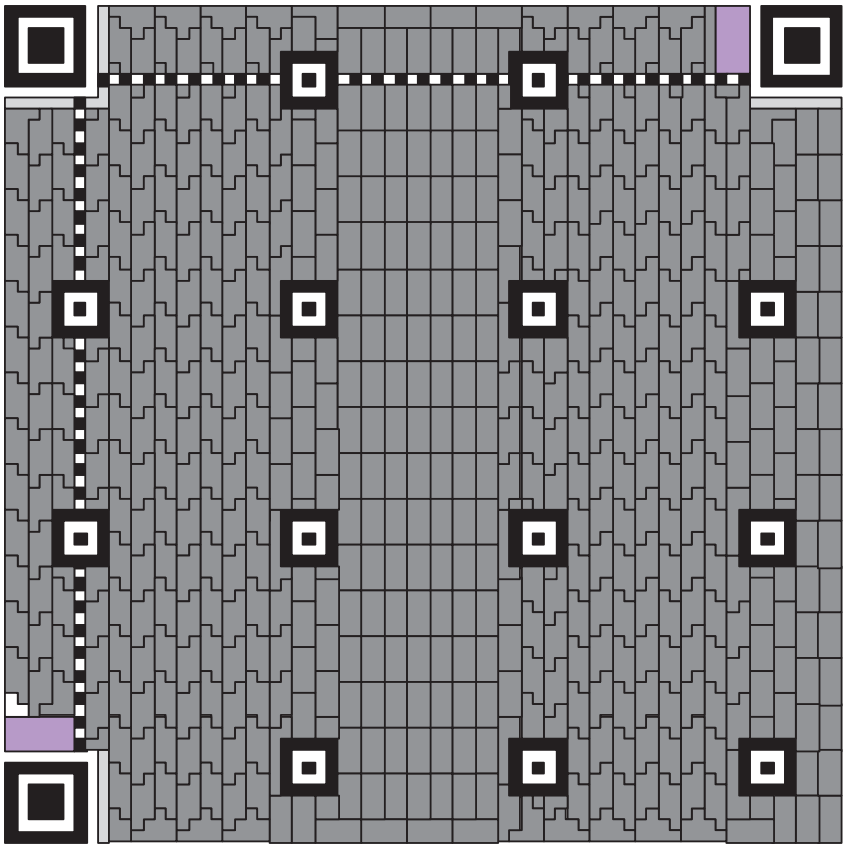
Obr. 10 - Struktura QR kódu verze 1



Obr. 11 - Struktura QR kódu verze 2

2.2.3 Velikosti QR kódu

QR kódy jsou několika velikostí (=verzí). Od verze 1, kdy matice je rozměru 21x21 bodů (Obr. 10), až po verzi 40, kdy má matice rozměr 177x177 bodů. Na Obr. 12 je ukázán jen menší kód verze 14 o rozměru 73x73 bodů, na kterém je dobře vidět, jak přibývají zarovnávací obrazce. Existuje také Micro QR kód, který je ve verzích M1-M4, kdy M1 se skládá z 11x11 bodů, M4 z 14x14 bodů. Micro QR kódy se tato práce nebude dále zabývat, protože nejsou tak používané v praxi a lze do nich uložit jen malý objem dat oproti klasickým QR kódům.



Obr. 12 - QR kód verze 14

Mód	Indikátor módu
ECI	**0111**
Numerický	**0001**
Alphanumerický	**0010**
Byte	**0100**
Kanji	**1000**
Structured Append (Strukturované spojování kódů)	**0011**
FNC1	**0101** (1. pozice) **1001** (2. pozice)
Koncový znak (konec zprávy)	**0000**

Tab. 2 - Indikátory datových módů

Verze	Numerický mód	Alphanumerický mód	Binární mód	Kanji mód
1 až 9	10	9	8	8
10 až 26	12	11	16	10
27 až 40	14	13	16	12

Tab. 3 - Počet bitů vyhrazených pro indikátor počtu znaků

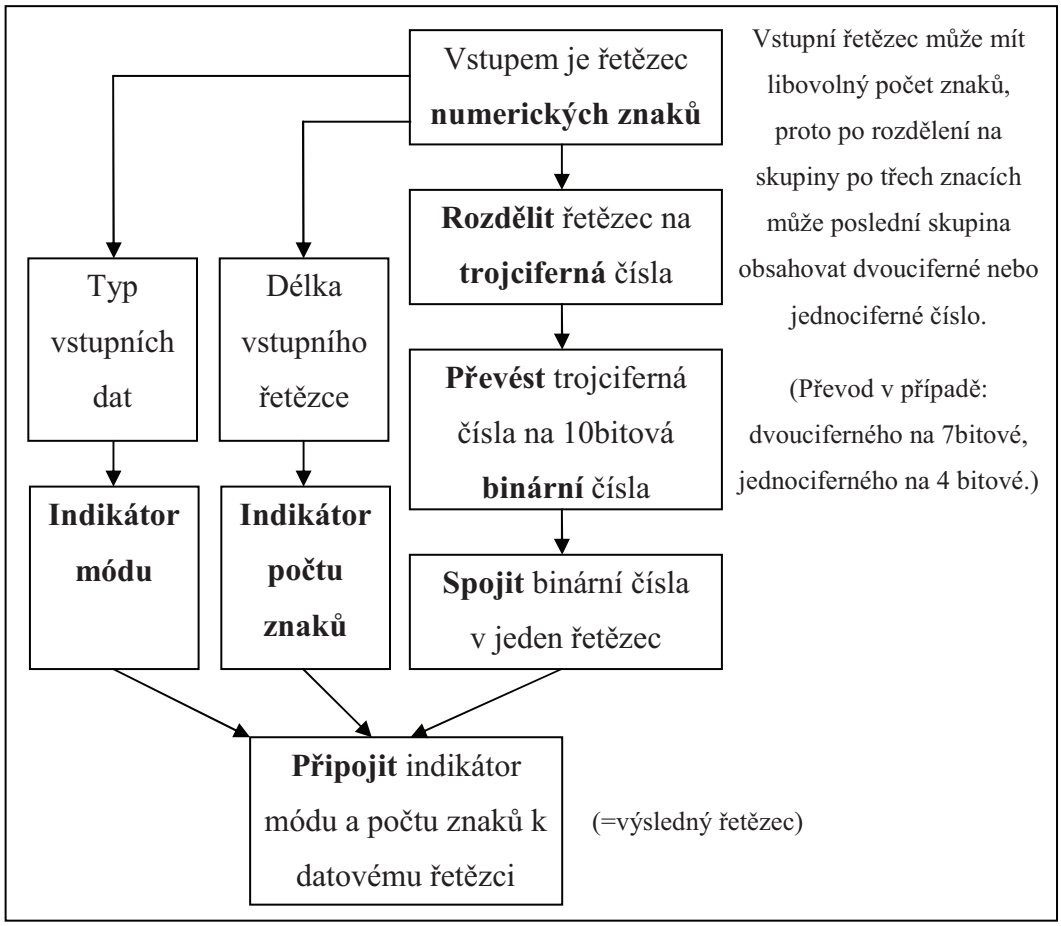
Písmeno	Hodnota	Písmeno	Hodnota	Písmeno	Hodnota
0	0	F	15	U	30
1	1	G	16	V	31
2	2	H	17	W	32
3	3	I	18	X	33
4	4	J	19	Y	34
5	5	K	20	Z	35
6	6	L	21	SP	36
7	7	M	22	$	37
8	8	N	23	%	38
9	9	O	24	*	39
A	10	P	25	+	40
B	11	Q	26	-	41
C	12	R	27	.	42
D	13	S	28	/	43
E	14	T	29	:	44

Tab. 4 - Tabulka znaků pro alfanumerické kódování

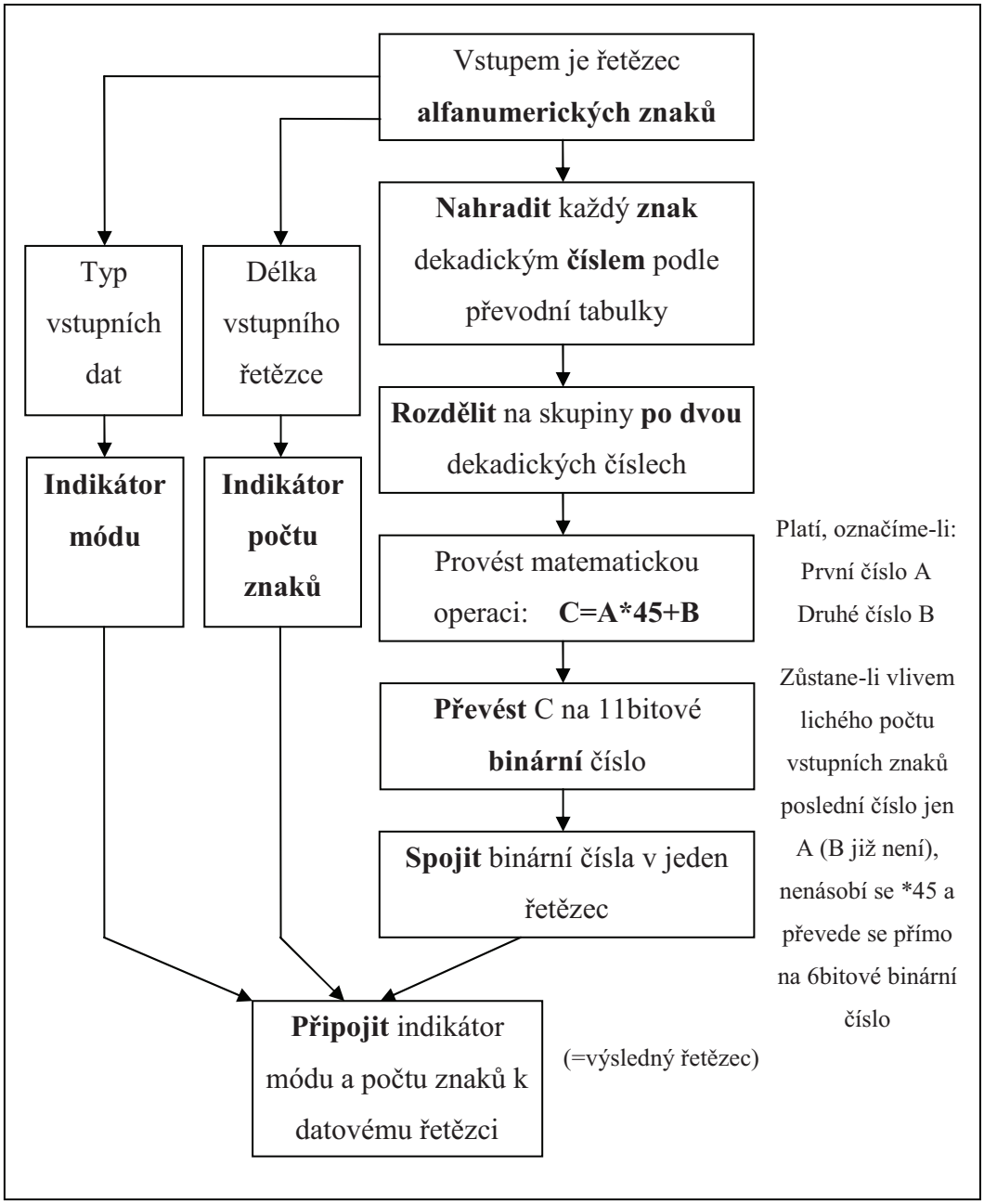
2.2.4 Kódování vlastních dat

Vlastní data nesená QR kódem mohou být několika módů: Extended Channel Interpretation (ECI), numerický, alfanumerický, binární a Kanji. Módy je možné i kombinovat. Na počátku je tedy nutné vybrat jaký datový mód budeme používat. Podle Tab. 2 se zvolí indikátor módu a podle verze QR kódu je odečteno v Tab. 3 kolik bitů, bude mít informace o počtu kódovaných znaků.

Dále je uveden postup převodu řetězce numerických a alfanumerických znaků na binární číslo se všemi náležitostmi. Pro numerické znaky je postup na Obr. 13 a pro alfanumerické na Obr. 14.



Obr. 13 - Postup převodu numerických znaků

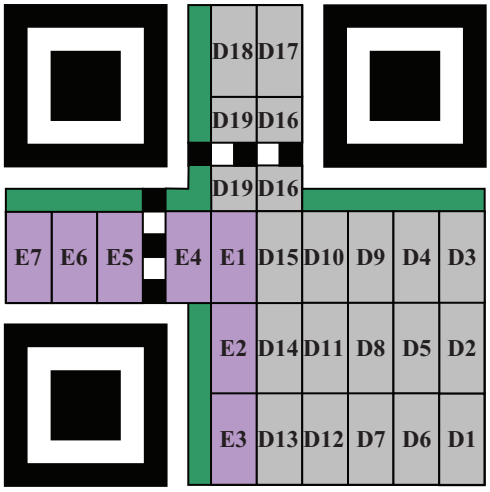


Obr. 14 - Postup převodu alfanumerických znaků

2.2.5 Rozdělení dat QR kódu na codewords

Codeword (CW) by se dalo přeložit jako zakódované slovo. Myšleno jako 8bitové binární číslo, které se v QR matici vyskytuje v přesně definovaném tvaru a velikosti. Velikost je odvozena od zmíněných osmi bitů, což znamená, že je to určitá oblast, ve které je 8bitová informace, kdy logickou jedničku představuje černý modul a logickou nulu bílý modul v této oblasti. Většinou má CW pravidelný tvar obdélníku, ale někdy je jeho tvar nepravidelný, například kvůli přítomnosti zarovnávacího symbolu. Jak to v tom případě vypadá je dobře vidět na Obr. 12.

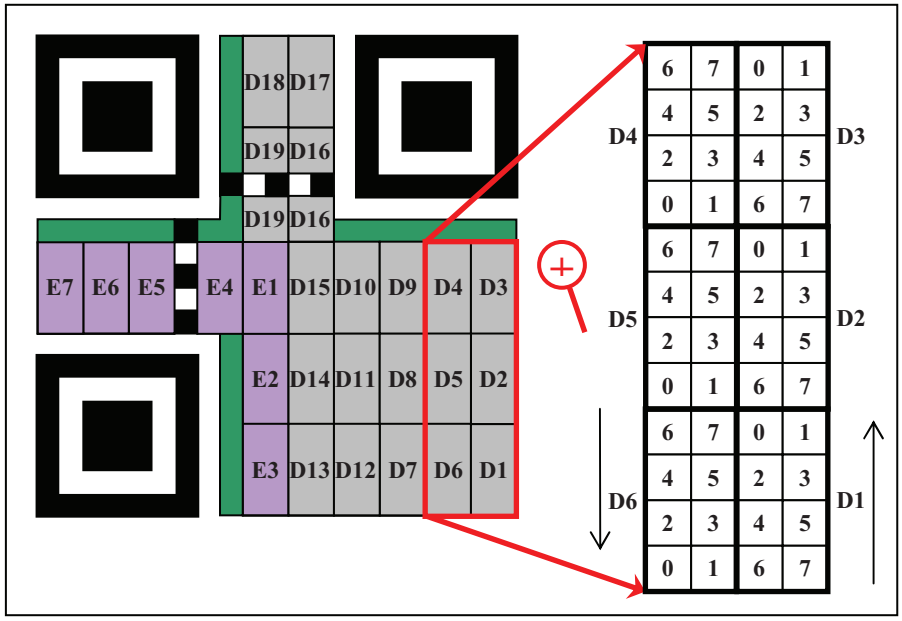
Na Obr. 15 je zobrazeno číslování datových CW (D1-D19) a CW korekce chyb (E1-E7) na QR kódu verze 1. Poměr počtu datových CW a CW korekce chyb je různý podle použité úrovně Reed-Solomonova kódování (více viz kapitola 2.2.6). Na tomto obrázku má QR kód verze 1 vždy celkem 26 CW. Zvolena byla korekce chyb úrovně L, a proto z celkového počtu obsahuje 7 CW korekci chyb.



Obr. 15 - Rozmístění codewords

Je však důležité přesně definovat i to, jak číst data z jednotlivých CW. Princip je zobrazen na Obr. 16. Záleží také na tom, kterým směrem roste číslo CW (D1-D19) Bit označený jako 7 je v případě růstu vzhůru v pravém dolním rohu, v případě růstu

dolů v pravém horním rohu. Potom se číslo bitu zvyšuje střídavě vlevo a vpravo jak je naznačeno na výše zmíněném obrázku. Bit 7 je nejvýznamnější bit (MSB).



Obr. 16 - Rozložení jednotlivých bitů v CW

2.2.6 Korekce chyb

QR kódy využívají pro korekci chyb Reed-Solomonovy algoritmy. Úrovně korekce chyb jsou L (7%), M (15%), Q(25%) a H(30%). Procenta v závorkách značí, kolik procent je možno obnovit při poškození QR matice. Zároveň se také mění poměr datových codewords a CW korekce chyb. Platí úměra, čím vyšší úroveň korekce chyb, tím vyšší počet CW korekce chyb a nižší počet CW použitelných pro vlastní data. V této práci jsme se omezili na práci s QR kódy verze 1, tedy velikosti 21x21 modulů. Tomu odpovídá celkem 26 CW. Poměr dat a korekce chyb u této verze pro různé úrovně shrnuje Tab. 5.

Pro každou velikost QR kódu existuje přesně definovaný poměr dat a korekce chyb, které je možné zjistit v [5]. Je to tabulka, která pro svoji rozsáhlost není v této práci celá uvedena. Tato práce se nezabývá Reed-Solomonovou korekcí chyb více do hloubky, a není zde tedy nijak popsán jejich přesný algoritmus.

V Tab. 6 je uveden mimo jiné také binární indikátor použité korekce chyb. Je důležitým prvkem při tvorbě i dekódování QR kódu a určuje jaká úroveň Reed-Solomonovy korekce chyb byla použita při tvorbě kódu.

Úroveň korekce chyb	Počet CW pro vlastní data	Počet CW pro korekci chyb
L	19	7
M	16	10
Q	13	13
H	9	17

Tab. 5 – Poměr datových CW a CW korekce chyb

Úroveň korekce chyb	Obnovitelnost kódu (%)	Binární indikátor
L	7	01
M	15	00
Q	25	11
H	30	10

Tab. 6 - Úrovně korekce chyb

2.2.7 Používané masky

Konečný krok při tvorbě QR kódu je provedení logické funkce XOR mezi maskou a dosud vytvářenou QR maticí. Ta se v tom okamžiku skládá z vlastních dat převedených do binárního kódu.

Masky jsou vzory o velikosti QR kódu, které slouží k tomu, aby ve výsledném QR kódu nevznikala souvislá místa černé nebo bílé barvy. Kdyby byl snímek s kódem pořízen obrazovým snímačem s nižším rozlišením, souvislá místa na pořízeném snímku by nemusel software správně rozpoznat. Proto existuje 8 masek, přičemž se vybere ta, která vytvoří výsledný obraz tak, aby tato spojitá místa nevznikala. Dále jsou popsány algoritmy, které dokáží vypočítat přesnou hodnotu tzv. trestných bodů za nevhodné rozmístění černých bodů. Vybírá se maska s nejmenším počtem trestných bodů.

V následující tabulce je zobrazen výpočet proměnných N_1 až N_4 představující trestné body pro určení nejvhodnější masky QR kódu. Čím menší jsou tyto proměnné tím je kód vhodnější. Za nerozlišitelné obrazce jsou považovány masky, kdy je vypočteno N_1=3, N_2=3, N_3=40, N_4=10.

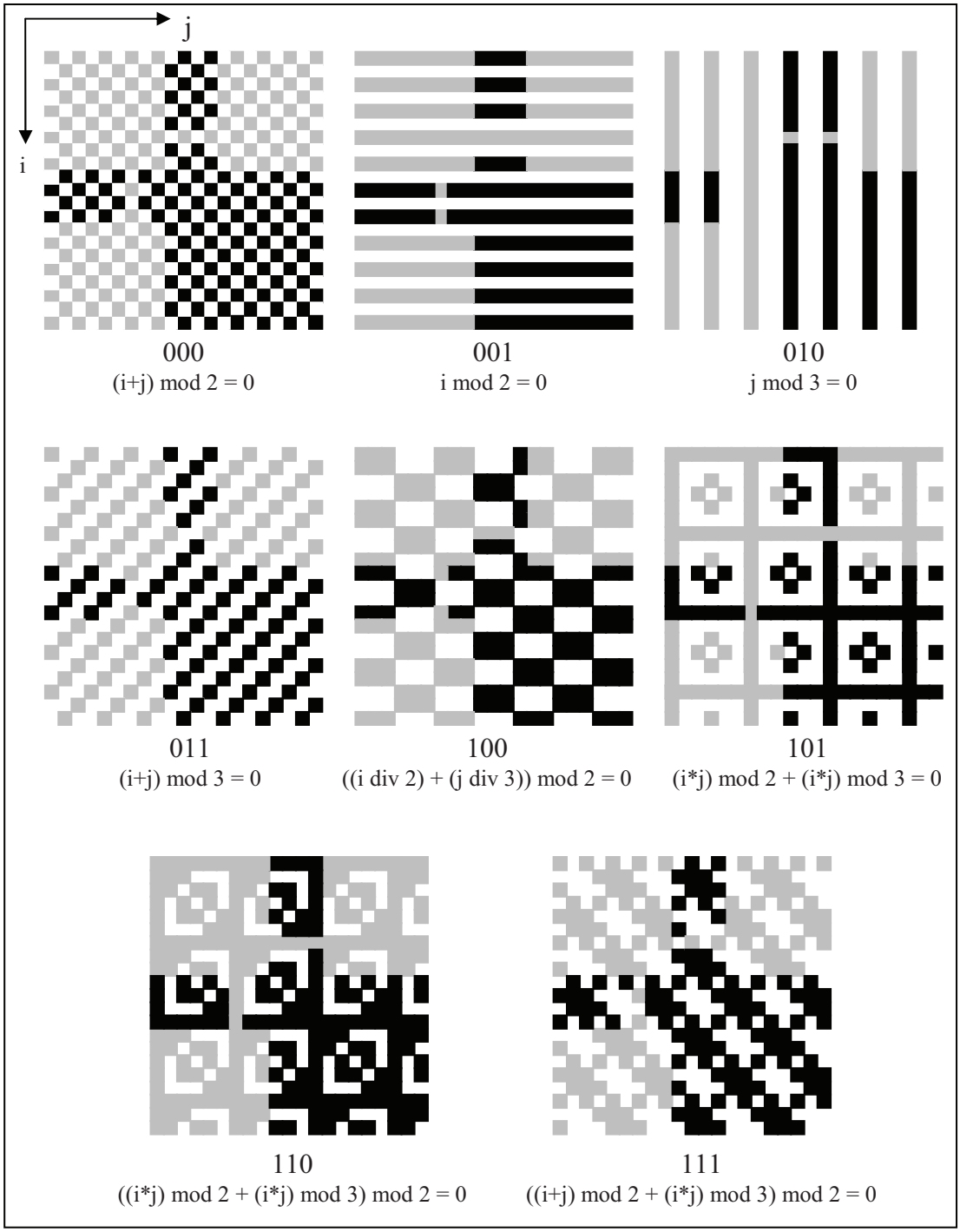
Jsou-li sousední moduly v řádku, či ve sloupci, stejné barvy, potom proměnná i udává počet modulů, které přesahují počet 5 modulů v této řadě.

Proměnná k značí o kolik procent (udáváno po 5%) se odchyluje počet černých modulů od ideálního 50% poměru černých a bílých modulů.

Předmět	Vyhodnocení podmínky	Body (trestné)
Sousední moduly v řádku nebo ve sloupci stejné barvy	Počet modulů = (5 + i)	N_1 + i
Blok modulů stejné barvy	Velikost bloku = m × n	$N_2 \times (m - 1) \times (n - 1)$
1 : 1 : 3 : 1 : 1 poměr (tmavý:světlý:tmavý:světlý:tmavý) obrazec v řádku/sloupci, následován bílou oblastí 4 moduly širokou	Existence obrazce	N_3
Poměr tmavých modulů v celém symbolu	$50 \pm (5 \times k)\%$ až $50 \pm (5 \times (k + 1))\%$	$N_4 \times k$

Tab. 7 – Vzorce pro výpočet nejvhodnější masky

Všechny užívané masky jsou zobrazeny na Obr. 17, kde šedě zbarvené části vzorů nejsou uplatněny při tvorbě výsledného kódu. Pod každým obrázkem je uvedeno binární číslo představující číslo masky a na dalším řádku je rovnice, která ukazuje jakým algoritmem je obrázek vytvořen.


j
i
000
(i+j) mod 2 = 0
001
i mod 2 = 0
010
j mod 3 = 0
011
(i+j) mod 3 = 0
100
((i div 2) + (j div 3)) mod 2 = 0
101
(i*j) mod 2 + (i*j) mod 3 = 0
110
((i*j) mod 2 + (i*j) mod 3) mod 2 = 0
111
((i+j) mod 2 + (i*j) mod 3) mod 2 = 0

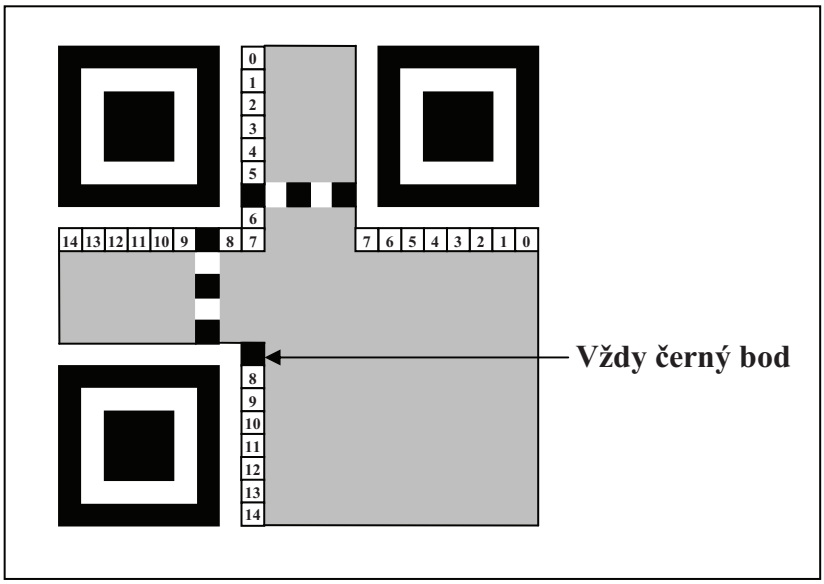
Obr. 17 - Masky

2.2.8 Kódování informace o použité korekci chyb a masce

Tyto data jsou uložena do míst na Obr. 10 označených zelenou barvou. Na Obr. 18 je znázorněno jak odečíst 15 bitové slovo nesoucí zmíněné informace z QR symbolu. Nejvýznamnější bit (MSB) odečítaného slova je na pozici čísla 14 v obrázku. Bit 14. a 13. obsahuje informaci o úrovni korekce chyb, 12., 11. a 10 bit udává použitou masku. Ostatních 10 bitů obsahuje korekci chyb typu BCH (15, 5), která pro svoji složitost není v práci více popsána.

Výše zmíněné 15 bitové slovo se však do kódu nevkládá přímo v této podobě. Je zapotřebí provést logickou operaci XOR mezi těmito daty a maskou, která je dle pravidel tvorby QR kódu dána: **101010000010010**. Výsledek je možno vložit do QR kódu na určené místo.

Příklad vstupních dat:	**111101011001000**
Maska:	**101010000010010**
Výsledek operace XOR:	**010111011011010**
	MSB LSB



Obr. 18 - Kódování informace o použité masce a korekci chyb

2.2.9 Příklad zakódování dat do QR kódu

V této kapitole je popsáno, jak vytvořit svůj kód krok po kroku. Vstupní data budou v alfanumerickém formátu. Výstupní kód bude verze 1, to znamená velikosti 21x21 bodů. Reed-Solomon kódování bylo zvoleno úrovně Q, tedy bude možné při poškození kódu obnovit až 25% dat. Zvolený řetězec k vložení do QR kódu je: **MARTINS**

1. Nahradit každý alfanumerický znak vstupního řetězce odpovídajícím desítkovým číslem dle Tab. 4.

Vstupní řetězec: **MARTINS**.
M = **22**, A = **10**, R = **27**, T = **29**, I = **18**, N = **23**, S = **28**.
MARTINS = (22,10,27,29,18,23,28)

2. Rozdělit na skupiny po dvou dekadických číslech z minulého bodu. Přičemž první číslo z dvojice je označeno A a druhé B (A,B)

(22,10) (27,29) (18,23) (28)

3. Provést matematickou operaci dle vzorce C = A*45 + B. V případě, že počet dekadických čísel byl lichý a zůstalo jako poslední jen jedno číslo, pak se toto číslo nenásobí číslem 45 a převádí se v dalším bodě přímo na 6bitové binární číslo.

C1 = 22*45 + 10 = **1000**
C2 = 27*45 + 29 = **1244**
C3 = 18*45 + 23 = **833**
C4 = **28**

4. Převést jednotlivá dekadická čísla do dvojkové soustavy. Kromě výjimky popsané v předešlém bodě, se převádí vždy na 11bitová čísla.

1000 = **01111101000**
1225 = **10011011100**
833 = **01101000001**
28 = **011100**

5. Spojit binární čísla v jedno

01111101000 10011011100 01101000001 011100

6. Vybrat indikátor módu dle Tab. 2.

Typ vstupních dat: alfanumerická data.
Indikátor módu: **0010**

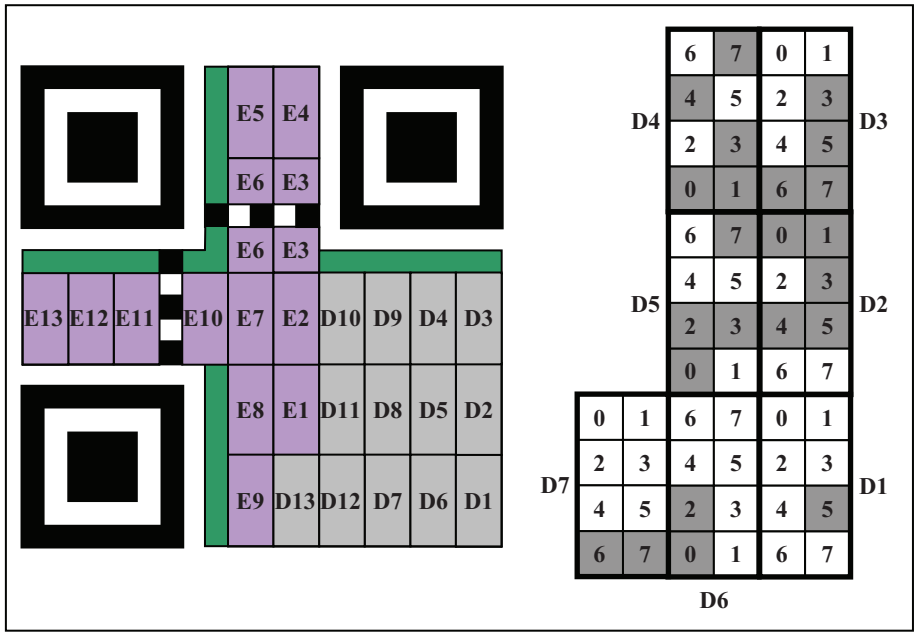
7. Vytvořit indikátor počtu znaků, jeho délka se řídí dle Tab. 3.

Počet znaků = 7
Počet znaků ve dvojkové soustavě (9bitů) = **000000111**

8. Spojit indikátor módu + indikátor počtu znaků + řetězec z bodu 5.

0010 000000111 01111101000 10011011100 01101000001 011100
Oddělení mezerami po 8 bitech (poslední bity doplněny nulami do 8bitů):
00100000 00111011 11101000 10011011 10001101 00000101 11000000
↑MSB **LSB↑**
Každých 8 bitů nyní představuje 1 CW. Celkem tedy data budou v 7 CW.

Protože byla zvolena korekce chyb úrovně L, bude poměr datových a korekčních CW tak, jak je na Obr. 19. Na tomto obrázku jsou také znázorněna přesně jednotlivá CW i s vypočítanými daty. Ze zdroje [5] bylo odečteno, že v tomto případě je 13 korekčních CW a 13 datových CW.

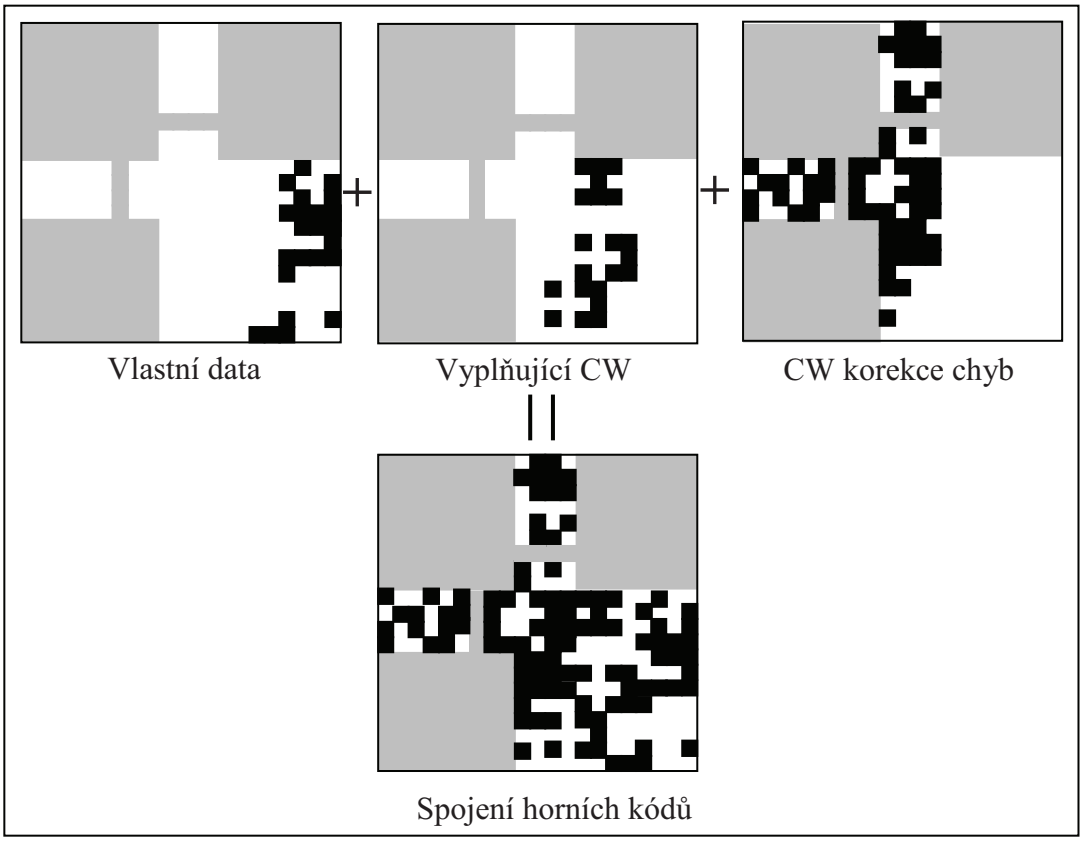


Obr. 19 - Příklad kódu - Rozložení CW

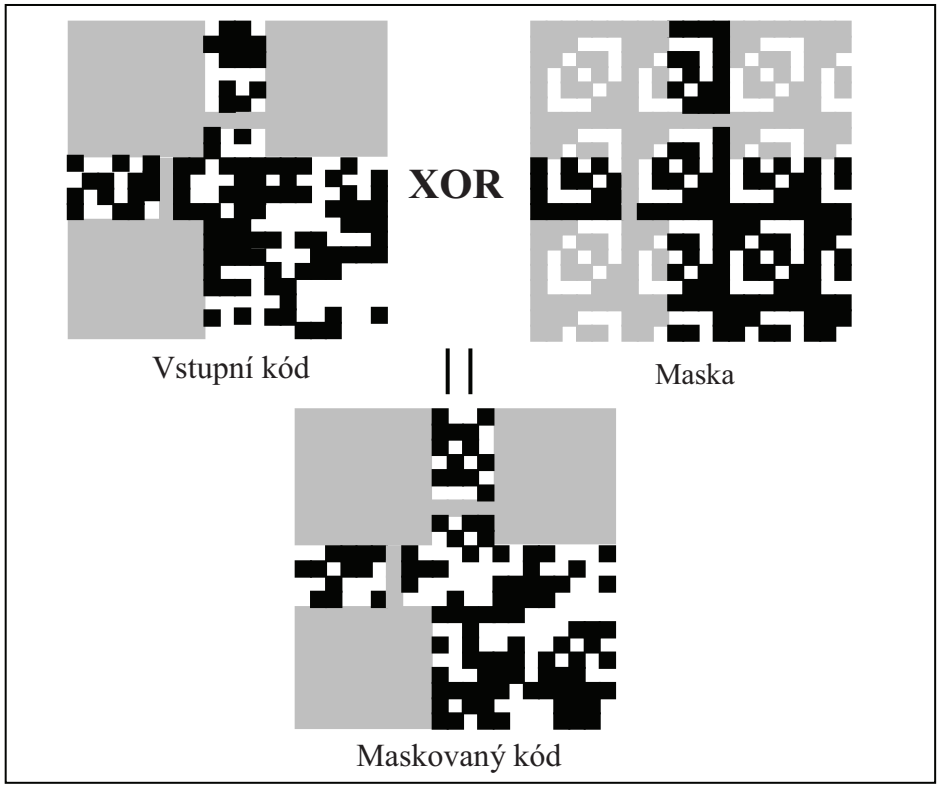
9. Do nevyužitých datových CW doplnit vyplňující CW

Na Obr. 20 vlevo jsou vytvořená CW vložena do kódu. Nevyužitá místa se vyplní tzv. Pad codewords - vyplňujícími CW. Jsou zobrazeny na obrázku uprostřed. Jejich tvar je: **11101100** v prvním CW a v následujícím CW je: **00010001**. Tyto dva CW se opakují až do úplného vyplnění datových CW kódu.

10. Na Obr. 20 vpravo jsou CW obsahující korekci chyb. Jak bylo uvedeno v kapitole 2.2.6, korekcí chyb se pro její složitost tato práce tolik nevěnuje. Dosavadní znalosti tedy zatím nedovolují přesně vygenerovat data korekce chyb, a proto byly získány z existujícího kódu odebráním masky. Kód byl vygenerován pomocí generátoru přístupného online na www stránkách, které jsou uvedeny jako zdroj [15].



Obr. 20 - Příklad kódu - Data, vyplňující CW a korekce chyb



Obr. 21 - Příklad kódu - Maskování kódu

Kódování informace o použité korekci chyb a masce

Zvolená korekce chyb je úrovně L, její indikátor = **11**

Nejvíce vyhovující maska (odečtena z existujícího vygenerovaného QR kódu): je číslo 6, její indikátor = **110.**

Tyto data jsou také zabezpečena pomocí korekce chyb. Její výsledek je vybrán z tabulky v literatuře [5] nebo vytvořen pomocí algoritmu BCH kódování = **1011001000.**

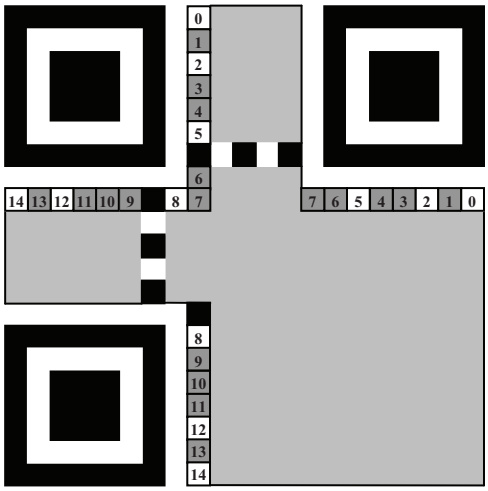
11 110 1011001000 = indikátor korekce chyb a masky + korekce chyb

10 101 0000010010 = maska (pro operaci XOR)

---

01 011 1011011010 = výsledné číslo připravené k vložení do QR kódu

MSB LSB



Obr. 22 - Příklad kódu - Informace o korekci chyb a masce

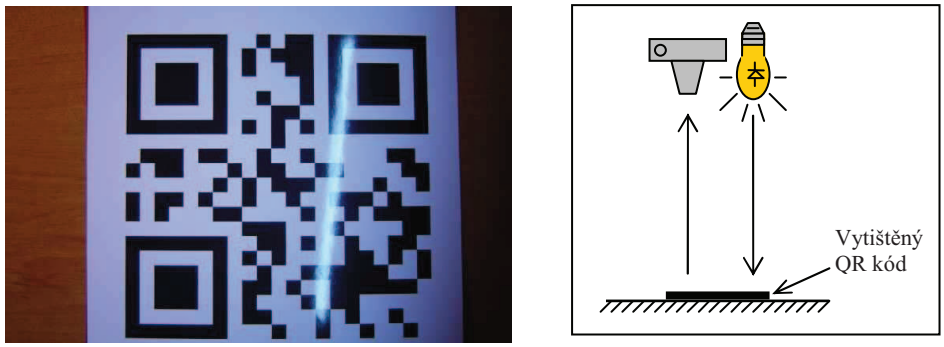
Obr. 23 - Příklad kódu - Výsledný QR kód v reálné scéně

3. POŘÍZENÍ DATABÁZE SNÍMKŮ

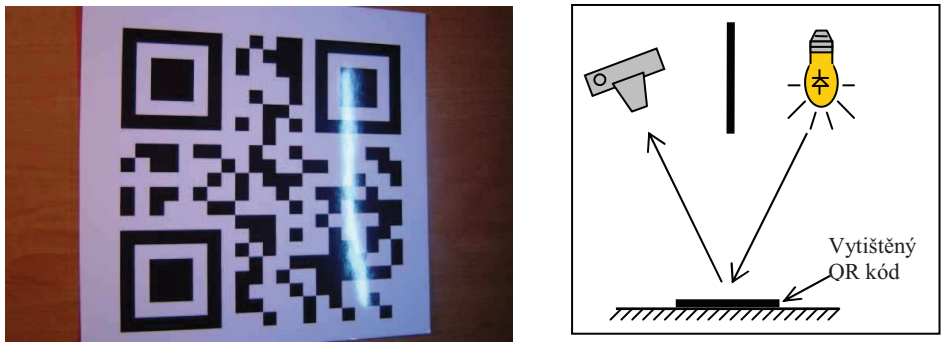
Bylo provedeno snímání několika reálných obrázků QR kódů. Pro snímání byly použity dva digitální snímače. Jedním z nich byl snímač zabudovaný do mobilního telefonu Sony Ericsson K750i, jehož rozlišení je 2Mpx (1632x1224px). Jako druhý byl použit digitální fotoaparát Sony DCS-P72 s rozlišením 2,8Mpx (2048x1360px). Jejich rozlišení je sice téměř stejné, ale optické zpracování obrazu v digitálním fotoaparátu je ještě před příchodem obrazu na CCD (CMOS) snímač značně kvalitnější, a tím i výsledná fotografie. Mobilní telefon disponuje jen jednoduchým optickým vybavením v omezeném prostoru, a tudíž jsou fotografie, obzvláště při špatně osvětlené scéně, méně kvalitní.

Reálný kód byl vyhotoven v několika velikostech na obyčejný bílý a na lesklý fotografický papír. Přisvětlování bylo prováděno pomocí žárovkového svítidla a LED svítidla (napájeného stejnosměrným napětím) ve tvaru trubice. Použity byly tři různé úhly svícení a snímání, jak je zobrazeno na obrázcích dále.

Na Obr. 24 je snímek lesklého QR kódu, kdy se LED svítidlo nachází na stejném místě nad scénou jako obrazový snímač. V lesklém povrchu je však vidět odraz svítidla stejně jako na Obr. 25, kde snímač a svítidlo svírají úhel přibližně 30°.

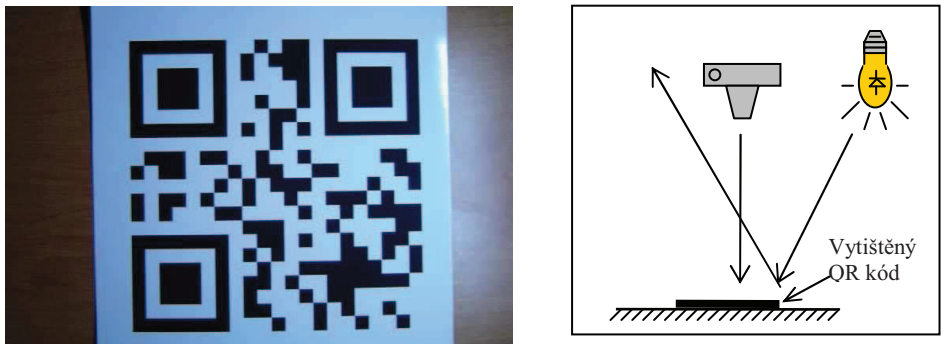


Obr. 24 - Snímek - digi. fotoaparát - lesklý kód - světlo LED shora



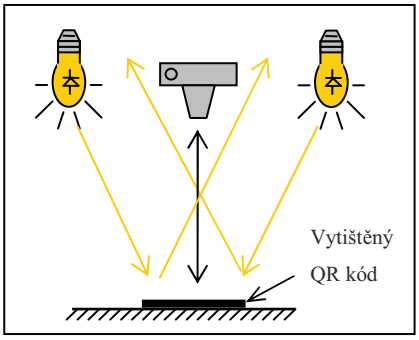
Obr. 25 - Snímek - digi. fotoaparát - lesklý kód - světlo LED šikmo

Jako nejlepší způsob snímání lesklého kódu byla zvolena scéna, kdy snímač je přímo nad QR kódem a světlo je bokem mimo osu okraje snímaného předmětu. Nevzniká tak odraz, který by poškodil kvalitu obrazu. Problém by mohl nastat v tom, že scéna není rovnoměrně osvětlena. To bude zřejmé až při zpracování obrazu.



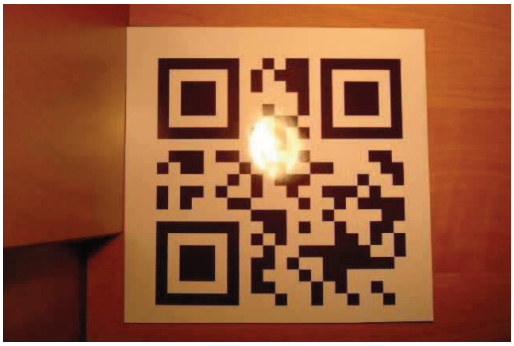
Obr. 26 - Snímek - digi. fotoaparát - lesklý kód - světlo LED bokem

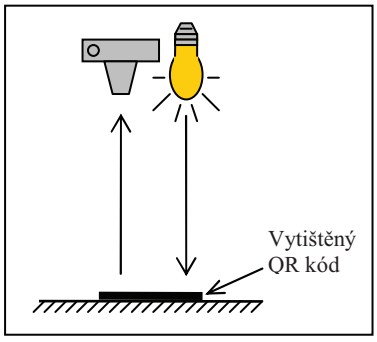
Dalo by se to vyřešit tím, že by se použily dva zdroje světla, přičemž každý z nich by byl na jiné straně snímaného předmětu, tak jak je na Obr. 27. Použitím těchto dvou nebo i více zdrojů se dosáhne rovnoměrně osvětlené scény. Snímání probíhá v ose snímaného předmětu, takže nedochází k deformacím obrazu.



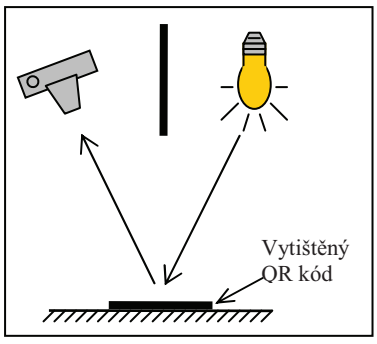
Obr. 27 – Nejvhodnější scéna pro pořízení fotografie

Dalším experimentem byla aplikace žárovkového světla. Postup byl stejný jako u LED svítidla. Byl také stejný výsledek, a to že nejlepší je umístit zdroj světla mimo půdorys QR kódu a snímač přímo nad kód.

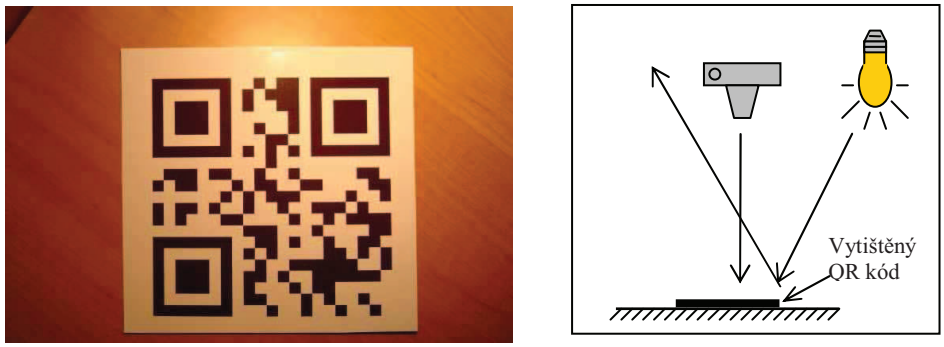




Obr. 28 - Snímek - digi. fotoaparát - lesklý kód - světlo žárovkové shora

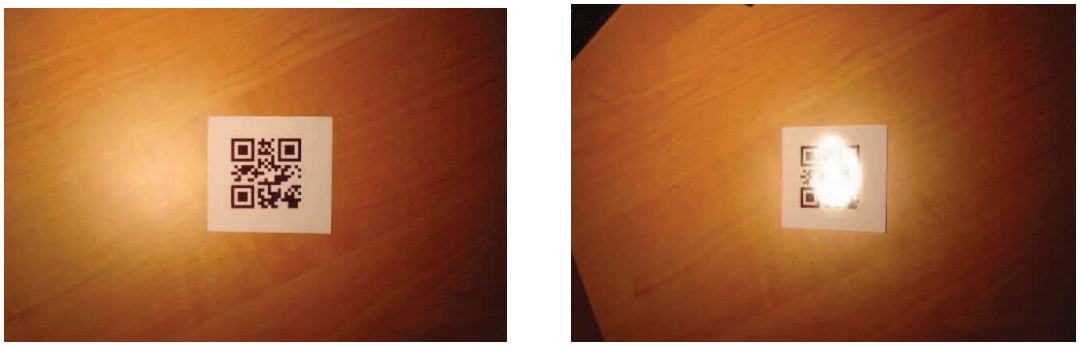


Obr. 29 - Snímek - digi. fotoaparát - lesklý kód - světlo žárovkové šikmo

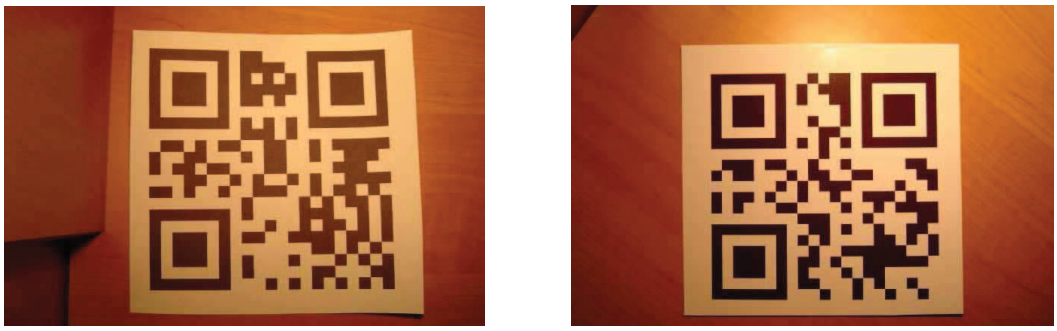


Obr. 30 - Snímek - digi. fotoaparát - lesklý kód - světlo žárovkové bokem

Na Obr. 31 je zobrazen QR kód vytištěný v menší velikosti, který je i přesto velmi dobře čitelný. Na druhém obrázku je vidět, jak moc může být poškozena jeho kvalita odrazem svítidla.



Obr. 31 - Snímek - digi. fotoaparát - malý lesklý kód - světlo žárovkové



Obr. 32 - Snímek - digi. fotoaparát - rozdíl mezi lesklým a matným

Při snímání mobilním telefonem byly snímky také kvalitní, ale při osvětlení žárovkovým svítidlem vznikaly přes obrázek vodorovné pruhy. Tyto byly pravděpodobně způsobeny postupným snímáním obrazu pomocí CMOS snímače ve vertikálním směru, kdy se při načítání obrazu projevuje kolísání jasu žárovky a případně další rušení.

I přes nízké rozlišení a snímání z větší vzdálenosti má kód na Obr. 33 (ve skutečnosti 3 cm velký) jeden bod (modul), který je schopen nést informaci, velikost 25x25 pixelů, což je pro zpracovaní obrazu více než dostatečné.

Obr. 33 - Snímek - mobilní tel. - malý matný kód - světlo žárovkové

Obr. 34 - Snímek - mobilní tel. - velký matný kód - světlo LED

Takže nejvýhodnější je kombinace snímače s vyšším rozlišením a dokonalejším optickým přizpůsobením a nasvětlení alespoň dvěma LED svítidly mimo půdorys snímaného QR kódu.

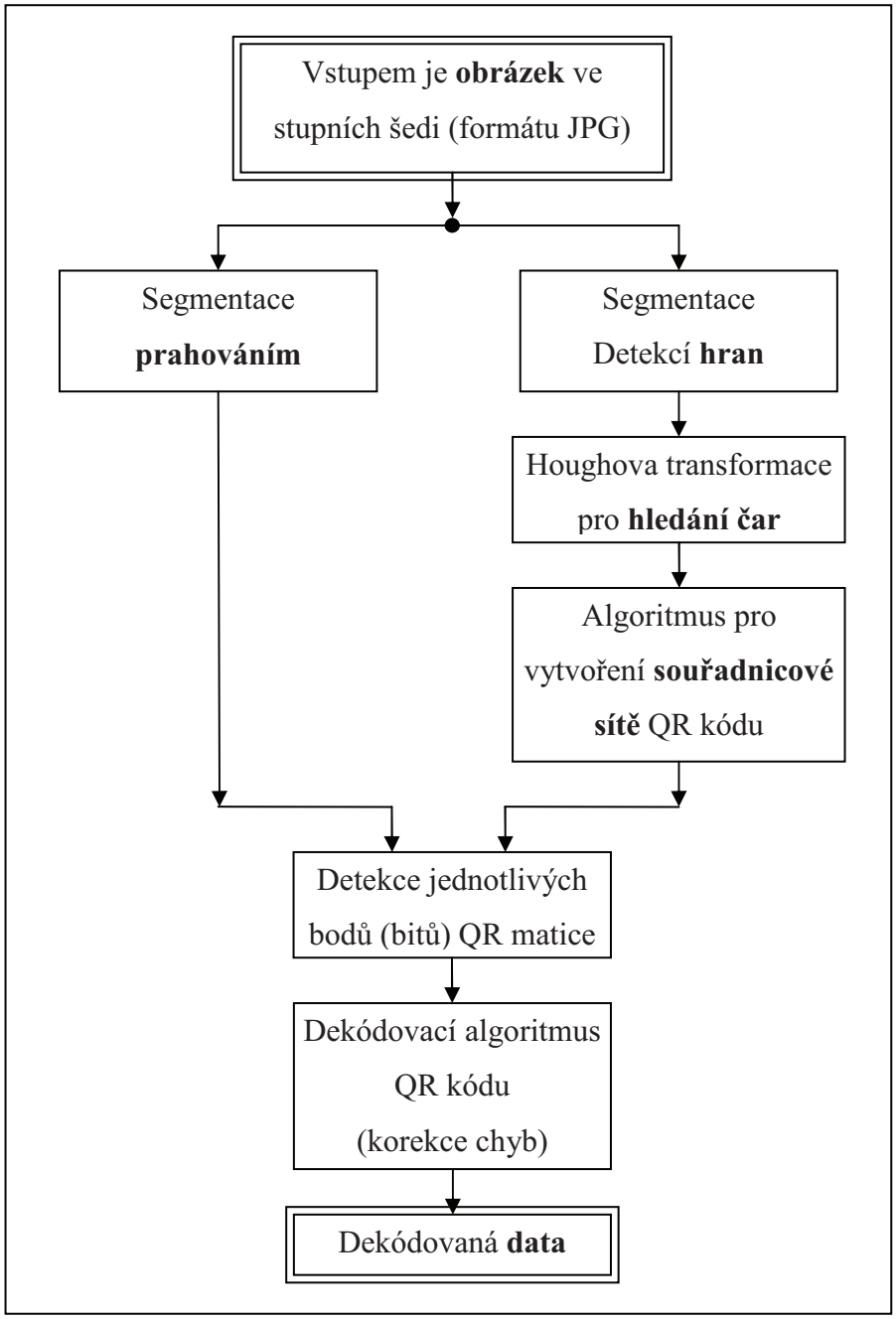
4. REALIZOVANÉ ŘEŠENÍ

Byl navrhnut algoritmus pro dekódování snímaného QR kódu, který byl oproti původnímu návrhu (v semestrální práci) změněn v několika prvcích a jeho pozměněná podoba je na Obr. 35. Nejdříve byl barevný vstupní obraz převeden při načítání do stupňů šedi, což zjednodušuje práci s ním a někdy je i obraz ve stupních šedi podmínkou funkčnosti některých funkcí knihovny OpenCV.

Dále bylo zjištěno, že vyprahovaný obraz není možné použít pro detekci hran, a proto se algoritmus nyní dělí na dvě větve. V jedné větvi se provede zmíněné prahování obrazu, ve kterém se později (až po zpracování pozice modulů v QR kódu) zjišťuje přítomnost černého nebo bílého modulu. Ve druhé větvi se v obrazu (ve stupních šedi) detekují hrany pomocí Cannyho detektoru, v těchto hranách se hledají čáry pomocí Houghovy transformace pro detekci čar.

Hledání obrysů nebylo dokončeno, k vytvoření algoritmu pro jejich hledání, by bylo zapotřebí více času. Z tohoto důvodu není možné kód přesně najít ve scéně, a je nutné zajistit samotný obraz QR kódu, aniž by se v okolí nacházely další prvky, které by se po aplikaci Houghovy transformace jevily jako čáry. Z tohoto důvodu nebylo možné vyřešit ani automatické otáčení kódu, a tudíž se předpokládá vstupní kód správně a přesně orientovaný.

Algoritmus je naprogramován v jazyce C++ s využitím knihovny pro zpracování obrazu OpenCV, která je popsána dále.



Obr. 35 – Návrh řešení – Algoritmus pro dekódování QR kódu

4.1 KNIHOVNA OPENCV

Název pochází z anglického Open source Computer Vision library. Je to tedy knihovna, jejíž zdrojové kódy jsou volně přístupné a je možné je upravovat. Je určena pro zpracování obrazu v reálném čase a práci s počítačovým viděním. Tato knihovna je napsána v jazyce C a C++ a běží pod operačním systémem Linux, Windows a Mac OS X. Je schopná využívat vícejádrové procesory.

Knihovna má 5 hlavních částí:

1. CXCORE – obsahuje datové struktury, maticovou algebru, správu paměti, datové transformace
2. CV – obsahuje zpracování obrazu, analýzu obrazových struktur, rozpoznání obrysů, kalibrace kamery
3. ML – Machine learning – funkce pro seskupování, klasifikaci a analýzu dat
4. HighGUI – poskytuje uživatelské rozhranní a funkce pro ukládání a volání obrázků či videí
5. CVCAM – obsahuje funkce pro práci s videi

4.2 POPIS JEDNOTLIVÝCH BLOKŮ NAVRŽENÉHO ŘEŠENÍ

Tato kapitola popisuje jednotlivé bloky navrženého řešení, které bylo uvedeno na Obr. 35.

4.2.1 Vstupní obrázek

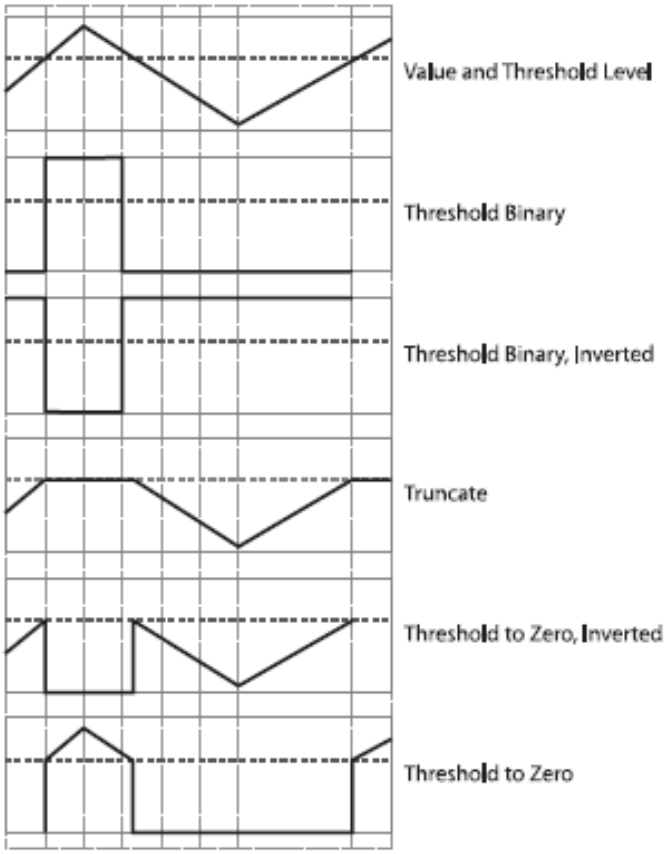
Vstupní obrázek má být ve formátu JPG, takže je možné použít přímo soubor vytvořený digitálním fotoaparátem. Obrázek by měl obsahovat QR kód o velikosti alespoň 400 x 400 pixelů (21 modulů o šířce alespoň 19 pixelů). Již ve funkci, která načítá obrázek do algoritmu v C++, se převede z barevného obrázku na obrázek ve stupních šedi (=grayscale). Činí se tak, protože nepotřebujeme informaci o barvách v obrazu a protože některé funkce knihovny OpenCV nepodporují barevný obraz.

Je důležité, aby vstupní obrázek obsahoval QR kód přibližně tak, jak je tomu na Obr. 36. Neměl by tedy být pootočený o více jak 1 – 2°. Kolem kódu by neměly být vytištěné žádné znaky, nečistoty nebo cokoliv, co by mohlo na vstupním obrázku představovat nějaký tmavý objekt. Tato podmínka platí především pro oblast nalevo a nahoře od QR kódu. Nebudou-li tyto podmínky dodrženy, bude špatně detekována souřadnicová síť QR kódu a dojde k jeho chybné detekci. Důvod této skutečnosti bude více popsán v kapitole 4.2.5.

Obr. 36 – Vstupní obrázek QR kódu

4.2.2 Prahování vstupního obrázku

V anglických textech je používáno označení Threshold. Prahování je proces, kdy se prochází hodnoty stupně šedi (0-255) ve všech pixelech obrazu. Při prahování se nastaví určitá hranice, při které se má nastavit výstup podle určité převodní funkce. Tuto převodní funkci je možné vybrat při použití funkce prahování (v OpenCV nazvána `cvThreshold`).



Obr. 37 – Typy převodních funkcí při prahování [4]

Cílem prahování v této práci bylo rozdělit obraz na oblasti, které jsou pouze černé a pouze bílé. Později, až bude známá souřadnicová síť a pozice, ve kterých se nachází moduly QR kódu, se bude snímat barva (černá/bílá) v tomto bodu. Proto byla použita převodní funkce `Threshold Binary` – to znamená, že pokud je aktuální

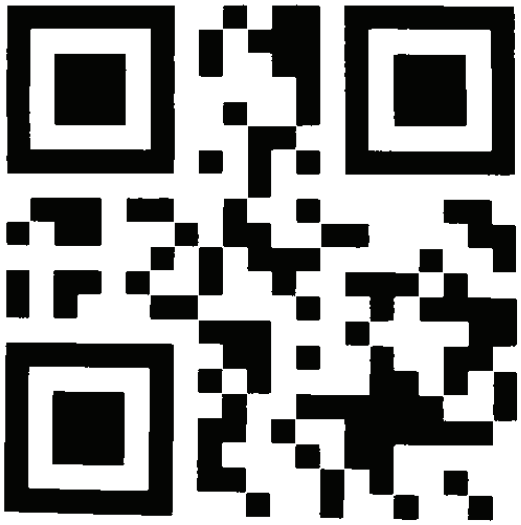
hodnota pixelu vyšší jak daná mez, nahradí se černou, je-li nižší, pak se nahradí bílou. Ostatní převodní funkce nejsou pro aplikaci v této práci příliš vhodné.

Příklad použití funkce `cvThreshold`:

```
cvThreshold( img1, img2, 140, 255, CV_THRESH_BINARY );
```

`img1` – vstupní obrázek
`img2` – výstupní obrázek
`140` – hodnota hranice pro prahování
`255` – maximální hodnota výstupu funkce prahování
`CV_THRESH_BINARY` – typ převodní funkce – binární (viz Obr. 37)

Výsledek funkce `cvThreshold` je tedy uložen do proměnné `img2`. Po jejím zobrazení pomocí funkce `cvShowImage( "Image_::2::", img2 );` se zobrazí obrázek takový, jako je ten na Obr. 38. Argument `"Image_::1::"` v příkazu představuje název okna, ve kterém se obrázek zobrazí. Při experimentech byly použity snímky rovnoměrně osvětlené pomocí dvou svítidel, takže byl výsledek po jednom prahování velmi dobrý. V případě nedokonale osvětleného snímku, by bylo možné použít prahování s plovoucím prahem.



Obr. 38 – Obrázek po prahování

4.2.3 Detekce hran v obraze

Pro detekci hran je používán Cannyho hranový detektor. Hrana se nachází v místě obrazu, kde se výrazně (v ideálním případě skokově) mění jas. Mění se tedy jasová funkce, a to se dá detekovat pomocí derivace. Cannyho detektor je zřejmě jedním z nejlepších hranových detektorů. Snaží se, aby hrany nechyběly a žádné se neopakovaly, na každou hranu reaguje pouze jednou a poloha hrany musí být dobře popsána.

Při použití Cannyho detektoru postupuje jeho algoritmus v těchto krocích:

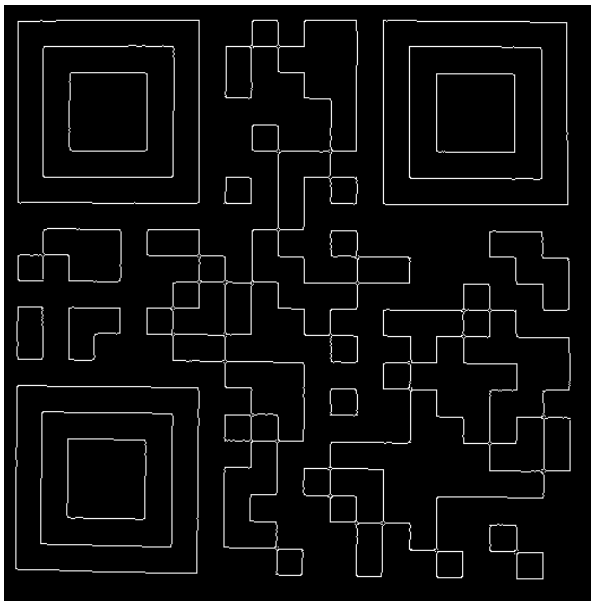
- eliminace šumu – Gaussovým filtrem
- nalezení hran – Sobelův filtr
- ztenčení – eliminace hran blízko sebe
- prahování – prahování s hysterezí, nastavení nižšího a vyššího prahu

Příklad použití funkce `cvCanny`:

```
cvCanny( img1, img3, 100 , 150 );
```

`img1 / img3` - vstupní / výstupní obrázek

`100` / 150 - hodnota nižšího / vyššího prahu pro prahování



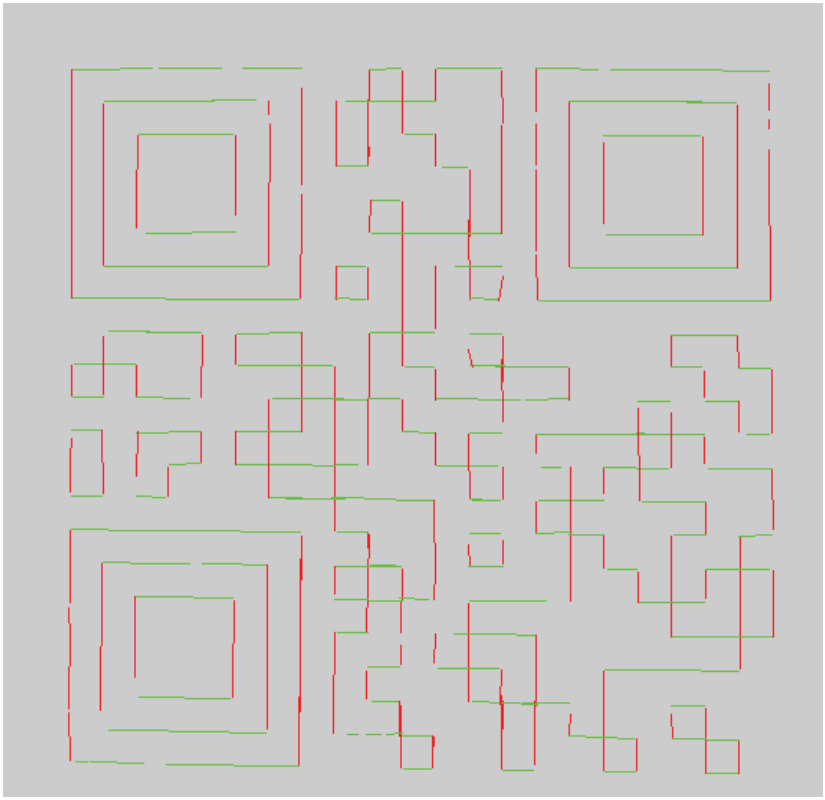
Obr. 39 – Obrázek po detekci hran

4.2.4 Houghova transformace pro detekci čar

Houghova transformace obecně slouží k detekci základních geometrických objektů v obraze (přímka, kružnice, elipsa, trojúhelník). V této práci bude výhodné hledat v obraze přímky.

Funkce z knihovny OpenCV, která zpracovává Houghovu transformaci pro hledání čar v obraze se označuje `cvHoughLines2`. Její výstup je ukládán do struktury o dvou prvcích (souřadnicích x a y). Ve struktuře za sebou je uložena vždy nejdříve souřadnice počátečního bodu a za ní souřadnice koncového bodu nalezené čáry.

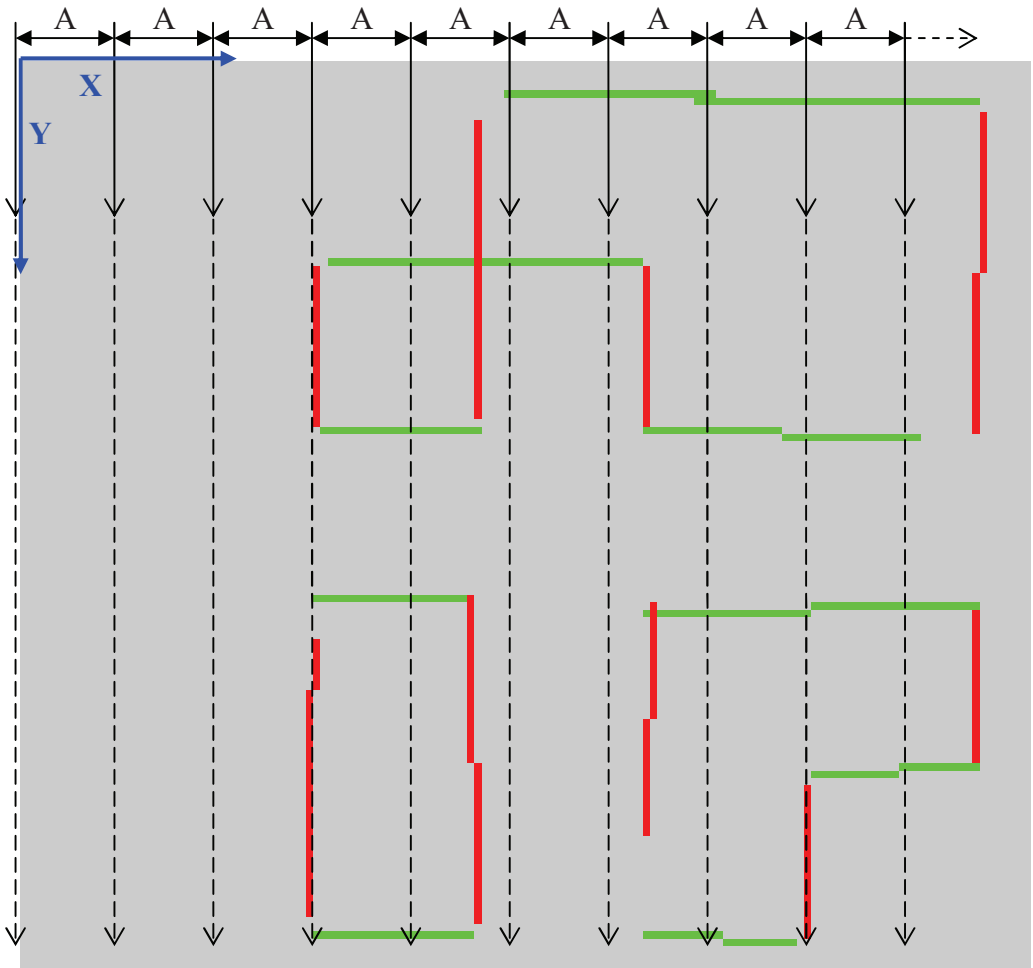
Při čtení těchto čar se zjišťuje, jestli je čára spíše svislá nebo vodorovná a podle toho se pro lepší orientaci nakreslí tak, jak je na Obr. 40, buď červenou nebo zelenou barvou. Zároveň je také vždy počáteční bod určité přímky uložen do pole `CarySvisle` nebo `CaryVodorovne` podle jejich orientace. Tato pole jsou dále zpracovávána.



Obr. 40 – Obrázek po Houghově transformaci

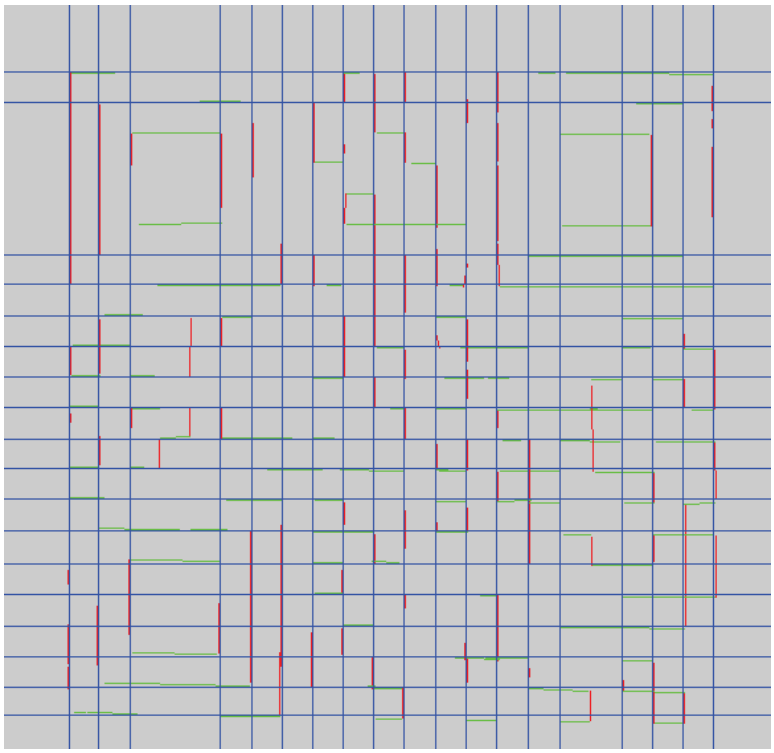
4.2.5 Algoritmus pro vytvoření souřadnicové sítě

Převezmou se pole `CaraSvisla` a `CaraVodorovna`, ve kterých jsou počáteční souřadnice čar, nalezených Houghovou transformací. Pro zjednodušení bude algoritmus vytvoření souřadnicové sítě popsán jen pro vertikální čáry. Pro tuto práci byla vytvořena funkce `Hough_aprox`. Nejdříve je obraz rozdělen do několika svislých oblastí o definované šířce, uložené v proměnné `sirka_oblasti`. Na Obr. 41 je tato šířka kótována symbolem A. V každé této oblasti se zjišťuje přítomnost svislých čar. To se provádí tak, že se hledá v proměnné `CaraSvisla` (typu pole - obsahující všechny počáteční body svislých čar) takový bod, který odpovídá svojí polohou prohledávané oblasti. Tyto se uloží do nového pole `Cara`.



Obr. 41 – Detail obrázku po Houghově transformaci

Pokud byl počet nalezených čar v oblasti vyšší jak 3, potom se uvažuje, že se v této oblasti opravdu nachází čára (není to šum) a spočítá se tedy aritmetický průměr polohy výsledného bodu (čáry). Tedy nejpravděpodobnější souřadnice čáry tvořící souřadnicovou síť QR kódu. Je-li toto provedeno v celé šířce obrazu, a to i v horizontální rovině s pomocí proměnné `CaraVodorovna`, vznikne obraz jako ten na Obr. 42, kde modré čáry představují aproximované čáry. Zde však vzniká omezení funkčnosti algoritmu, nachází-li se nalevo nebo nahoře od QR kódu nějaká čára. Ta může být způsobena nějakým šumem, okrajem nálepky QR kódu na předmětu, či nějaká hrana na předmětu samotném. To pak způsobí, že se začne souřadnicová síť vytvářet již od této nežádoucí čáry. Proto musí být nalevo a nahoře QR kódu prázdná plocha.



Obr. 42 – Obrázek s předběžnou sítí čar

Pro tuto aproximaci krátkých čar, které jsou různě posunuty vlivem malého nežádoucího pootočení nebo geometrickou deformací obrazu, byla vytvořena funkce

s názvem `Hough_aprox`. Jde totiž v podstatě o aproximaci čar nalezených Houghovou transformací. Tato funkce je volána s parametrem `CaraSvisla` a hned po té s parametrem `CaraVodorovna`. Jsou tak vypočteny aproximované souřadnice původních čar, které se uloží pomocí návratové hodnoty z funkce `Hough_aprox` do polí `CaraX` a `CaraY`.

Příklad použití funkce `Hough_aprox`:

```
CaraX = Hough_aprox(CarySvisle, pocetX, img4, 'x');
```

`CaraX`	-	pole, do kterého se má uložit návratová hodnota, která představuje pozici aproximovaných čar
`CarySvisle`	-	vstupní pole obsahující počáteční souřadnice všech čar
`pocetX`	-	počet aproximovaných čar v proměnné `CaraX`, je zároveň také návratovou hodnotou
`img4`	-	obrázek, do kterého se mají zakreslit aproximované přímky
`'x'`	-	udává, ve které ose se ve skutečnosti právě aproximuje, určuje hlavně v jakém směru se mají vykreslit aproximované čáry

Při popsaných předchozích operacích však může dojít k nežádoucím chybám. Pro jejich eliminaci byla vytvořena funkce `Cary_selektovani`, která funguje následujícím způsobem. Funkce si převezme jako parametr pole `CaraX (CaraY)`, ve kterém jsou uloženy souřadnice aproximovaných čar. Spočítá se vzdálenost mezi nimi a uloží se do nového pole `deltaX_Y` (univerzální název pole, protože se funkce `Cary_selektovani` volá nejdříve se souřadnicemi X a později se souřadnicemi Y). Z těchto vzdáleností se vypočítá průměr (`prum`) , který však není přesným průměrem vzdáleností dvou čar, je totiž ovlivněn mezerami příliš velkými nebo malými vlivem chybné aproximace.

Obr. 41 je vidět jeden z problémů, který často nastává. Mezi třetí a čtvrtou oblastí dochází k dělení svislé čáry, která spadá částečně do třetí oblasti a částečně do

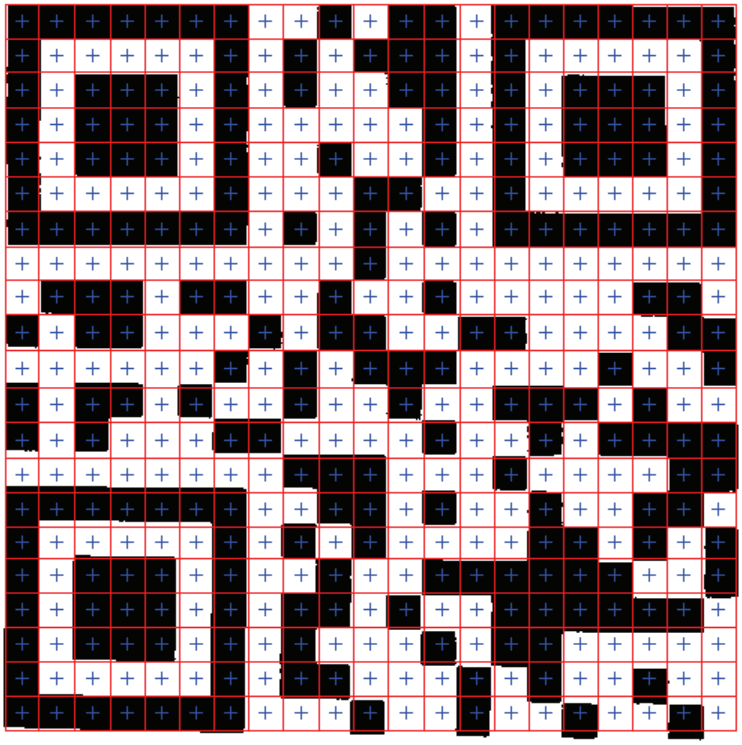
čtvrté. Je logické, že skutečná pozice čáry je na hranici těchto oblastí. Avšak algoritmus vytvoří dvě aproximované čáry blízko sebe (v každé oblasti jednu). Tento případ je detekován, pokud je hodnota vzdálenosti čar (`deltaX_Y`) menší než hodnota průměru snížená o hysterezi. Hystereze je zvolena pětina průměru `prum`.

Je-li uvedený případ zjištěn, potom je zapotřebí umístit novou čáru mezi tyto dvě, které byly chybným vyhodnocením funkce `Hough_aprox` umístěny blízko sebe. Je tedy vypočtena pozice mezi nimi, kam se přesune jedna z nich a druhá se odstraní.

Další chyba funkce `Hough_aprox` může spočívat v tom, že není dostatek údajů (počtu krátkých čar) nalezených Houghovou transformací pro hledání čar, a proto čára chybí v místě, kde má být. To je dobře vidět na obrázku Obr. 42, kde vznikly touto chybou prázdné proluky.

To detekuje funkce `Cary_selektovani` tak, že najde hodnotu vzdálenosti mezi čarami větší, než je hodnota průměru zvýšená o hysterezi, která je zvolena pětina průměru. V tom případě se tato vzdálenost z pole `deltaX_Y` odstraní. Mohla by se nahradit hodnotou průměru, ale protože průměr v tuto chvíli není přesný (je zkreslený chybně aproximovanými přímkami, které se tímto právě eliminují), není tak učiněno.

Po odstranění této hodnoty (příliš vzdálených přímek) z pole `deltaX_Y` a opravě polohy přímek, které byly příliš blízko, již obsahuje pole jen pravdivé vzdálenosti. Tyto odpovídají skutečné vzdálenosti čar souřadnicové sítě QR kódu. Z nich se tedy vypočítá přesný průměr. Je-li průměr desetinné číslo, zaokrouhlí se nahoru, což bylo experimentálně zjištěno jako výhodnější. Pomocí této nové průměrné hodnoty vzdálenosti přímek se provede doplnění chybějících přímek do celkového počtu 22 přímek, tzn. 21 modulů QR kódu. Funkce `Cary_selektovani` tedy vrací výsledné souřadnice sítě QR kódu. Ta je vykreslena na obrázku Obr. 43 červenou barvou.



Obr. 43 – Obrázek s vyznačením souřadnicové sítě

Když je vypočtena souřadnicová síť, dopočítá se střed každého oka této sítě a uloží se do polí `BodyX` a `BodyY`. Vzniknou tak souřadnice bodů, ve kterých budeme později zjišťovat přítomnost černého nebo bílého modulu QR kódu. Pro lepší orientaci a kontrolu dosavadních algoritmů jsou tyto body vyznačeny na Obr. 43 modrým křížkem s využitím vytvořené funkce `nakresli_krizek`.

Příklad použití funkce `nakresli_krizek`:

```
nakresli_krizek(BodyX[x], BodyY[y], img5);
```

`BodyX[x]`	-	bod na ose x (roste na obrázku zleva doprava), ve kterém má být křížek
`BodyY[y]`	-	bod na ose y (roste na obrázku shora dolů), ve kterém má být křížek
`img5`	-	obrázek, do kterého se má zakreslit křížek

4.2.6 Algoritmus pro zpracování matice QR kódu

Je-li vypočtena pozice modulů QR kódu, může se vytvořit matice logických 1 a logických 0. To se provede v cyklech pomocí funkce `cvGet2D`, která zjistí úroveň šedi (0-255) v daném bodě obrázku. To je potom následně převedeno na logické nuly a jedničky tak, že černá barva představuje logickou 1. Vytvoří se tak dvourozměrné pole `matice[21][21]`. Na Obr. 44 je zobrazena tato matice po výpisu to konzolového okna.

```
1 1 1 1 1 1 1 0 0 1 0 1 1 0 1 1 1 1 1 1 1
1 0 0 0 0 0 1 0 1 0 1 1 1 0 1 0 0 0 0 0 1
1 0 1 1 1 0 1 0 1 0 0 1 1 0 1 0 1 1 1 0 1
1 0 1 1 1 0 1 0 0 0 0 0 1 0 1 0 1 1 1 0 1
1 0 1 1 1 0 1 0 0 1 0 0 1 0 1 0 1 1 1 0 1
1 0 0 0 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 0 1
1 1 1 1 1 1 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 1 1 1 0 1 1 0 0 1 0 0 1 0 0 0 0 0 1 1 0
1 0 1 1 0 0 0 1 0 1 1 0 0 1 1 0 0 0 0 1 1
0 0 0 0 0 0 0 1 0 1 0 1 1 1 0 0 0 0 1 0 0 1
1 0 1 1 0 1 0 0 1 0 0 1 0 0 1 1 1 0 1 0 0
1 0 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 1 1 1 1
0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 0 1 1
1 1 1 1 1 1 1 0 0 1 1 0 1 0 0 1 0 0 1 1 0
1 0 0 0 0 0 1 0 1 0 1 0 0 0 0 1 1 0 1 0 1
1 0 1 1 1 0 1 0 0 1 0 0 1 1 1 1 1 1 0 0 1
1 0 1 1 1 0 1 0 1 1 0 1 0 0 1 1 1 1 1 1 0
1 0 1 1 1 0 1 0 1 0 0 0 1 0 1 1 0 0 0 0 0
1 0 0 0 0 0 1 0 1 1 0 0 0 1 0 1 0 0 1 0 0
1 1 1 1 1 1 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0
```

Obr. 44 – Výpis do konz. okna Binární matice – přečteno z obrázku

Při dekódování dat z QR kódu je zapotřebí nejdříve přečíst informaci o použité korekci chyb a použité masce. O typu použité masky je důležité vědět, aby mohla být z kódu odebrána a získala se tak matice vlastních dat. To se odečte z matice v těch místech, která byla popsána v kapitole 2.2.8. K tomu byla vytvořena funkce s názvem `maskaAkorekcechyb`. Tato funkce vyhledá v celé matici QR kódu jen ty moduly, které obsahují danou informaci. Ta je také překrytá maskou, která je však již definována a je možné ji z těch dat odebrat. Pak stačí jen odečíst použitou masku pro celý zbytek QR kódu a korekci chyb. Za těmito dvěma

informacemi se také nachází korekce chyb těchto dvou informací, která není v této práci více pospáno tak jak bylo řečeno v kapitole 2.2.8.

Příklad použití funkce `maskaAkorekcechyb`:

```
maska_korekce = maskaAkorekcechyb (matice);
```

`maska_korekce` - návratová hodnota funkce, první prvek tohoto pole obsahuje úroveň korekce chyb, druhý použitou masku

`matice` - tato matice odečtená z QR kódu je vstupem

Úroveň použitého Reed-Solomonova kódování je pouze vypsána do konzolového okna a je z čísla převedena na písmena, použitá k označování této úrovně. Korekce chyb není více využíváno, tak jak bylo uvedeno v kapitole 2.2.6.

Je-li známý typ použité masky. Vytvoří se maska podle vzorce daného typu masky a uloží do matice `maska`. Je také zobrazena v konzolovém okně. Pak vypadá výpis tak, jako je na Obr. 45.

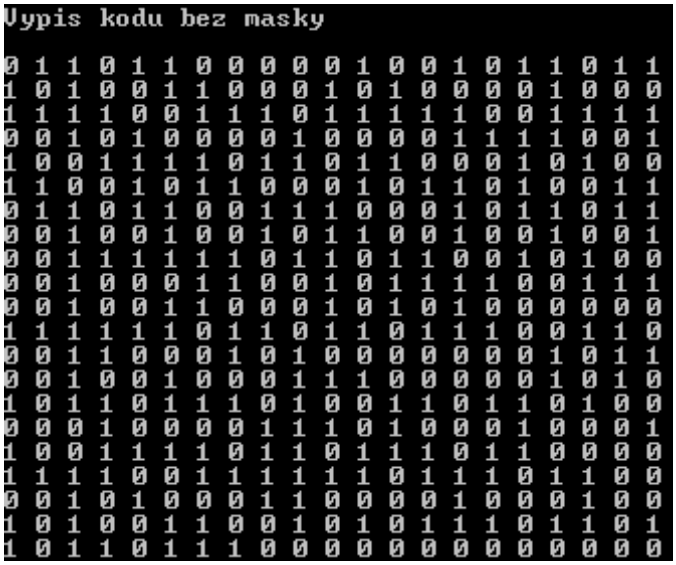
```
Detekovana korekce chyb je cislo:  3
Korekce chyb je urovne M


Detekovana maska je cislo:  3
Vypis masky

1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0
1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0
1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0
1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0
1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0
1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0
1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0
0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0
```

Obr. 45 - Výpis do konz. okna – Binární matice – maska, korekce chyb

Je-li známá matice odečtená z QR kódu a matice obsahující masku, mohou se pomocí operace XOR mezi těmito maticemi získat vlastní data. V této chvíli dojde ke zjednodušení tím, že by se správně měla tímto způsobem odebrat maska jen v místech vlastních dat a dat korekce chyb. Tudíž kotvící obrazec (soustředné čtverce), zaměřovací značky a místo, kde je uložena informace o použité masce a korekci chyb, by neměla být odebráním masky změněna. Pro zjednodušení algoritmu, k této změně však dochází. Tento postup však není nijak chybný, a neztrácíme tak žádnou informaci. Příklad matice bez masky je na Obr. 46.



Obr. 46 - Výpis do konz. okna – Binární matice – matice bez masky

Než se začnou z matice vlastních dat odečítat data, je nutné zjistit jaký typ dat je v QR kódu uložen. Tato informace se zjistí z prvních 4 bitů dat. Popisovaný algoritmus je omezen jen na alfanumerický datový typ, a to z důvodu velké složitosti a nepřehlednosti při zakomponování všech používaných typů.

Následujících 9 bitů je vyhrazeno pro uložení počtu znaků. To určí, kam až sahají data, a kde pokračují vyplňující CW. S těmito znalostmi je tedy možné začít odečítat vlastní data z QR kódu. K tomu byla vytvořena funkce `odecti_jen_data`.

Příklad použití funkce `odecti_jen_data`:

```
data = odecti_jen_data (kod_bezmasky);
```

`data`	-	návratová hodnota funkce, pole obsahující sekvenci logických nul a jedniček v přesném pořadí
`kod_bezmasky`	-	tato matice vlastních dat je vstupem do funkce

Pak je tedy zapotřebí dopočítat z počtu znaků, kam sahají data. Je také nutné zjistit jestli je počet znaků sudý nebo lichý. Je-li sudý, potom jsou vždy dvojice znaků uložené v 11 bitovém prostoru, a to tak, že první znak je násoben číslem 45 a druhý je k tomu přičten. Je-li počet lichý, potom jsou všechny znaky zakódovány do 11 bitových polí, jak bylo popsáno, ale poslední znak (lichý) je uložen do 6 bitů velkého prostoru a není násoben žádným číslem. Jeho hodnota přímo odpovídá určitému znaku. K rozlišení těchto stavů slouží proměnné `bit6` a `bit11`. S těmi se potom volá funkce `data_na_znaky`, která provádí výše popsané operace k získání celočíselných hodnot každého znaku.

Příklad použití funkce `data_na_znaky`:

```
znaky = data_na_znaky (data, bit11, bit6);
```

`znaky`	-	návratová hodnota funkce, pole obsahující celočíselné hodnoty jednotlivých znaků
`data`	-	pole obsahující sekvenci logických nul a jedniček v přesném pořadí – zakódované znaky
`bit6 a bit11`	-	pomocné proměnné k dekódování (lichost / sudost)

Nyní už jen stačí převést každý znak v celočíselné podobě na alfanumerický tvar podle tabulky v [5]. Nejedná se však o ASCII tabulku. Tuto tabulku využívá funkce vytvořená pro účel tohoto převodu. Jmenuje se `vypsat_znak` a zároveň také vypisuje znak do konzolového okna. Vizualizace potom může odpovídat Obr. 47.

Příklad použití funkce `vypsat_znak`:

```
vypsat_znak (znaky[i]);
```

`znaky` - pole obsahující celočíselné hodnoty jednotlivých znaků

```
Detekovany indikator modu je:  2
To znamena alfanumericky

Detekovany indikator poctu znaku je:  4


Znak 0: 10
Znak 1: 17
Znak 2: 24
Znak 3: 19

Dekodovane znaky:  AHOJ
```

Obr. 47 - Výpis do konz. okna – Dekódované znaky

5. ZHODNOCENÍ VÝSLEDKŮ REALIZOVANÉHO ŘEŠENÍ

Realizované řešení bylo vyzkoušeno na testovacích snímcích. Zhodnocení je nutné rozdělit do dvou kategorií. Jedna z nich je testování při dodržených podmínkách pro dekódování a druhá je při nedodržení těchto podmínek.

Zmíněné podmínky jsou:

- velikost vlastního QR kódu na snímku alespoň 400x400 pixelů
- QR kód správně orientovaný, pootočení maximálně 1-2°
- nalevo a nahoře od QR kódu nemají být rušivé objekty (kap. 4.2.5)
- v QR kódu nesmí být odraz svítidla, který by poškodil čitelnost
- snímek nesmí být geometricky deformován

Testování bylo prováděno na 6 snímcích vyhovujících podmínkám a 6 snímcích nevyhovujících podmínkám tak, jak ukazuje Tab. 8. Mezi snímky, u kterých proběhlo dekódování korektně, byly i ty, které měly například mírnou deformaci způsobenou snímáním zblízka či nepřesně kolmo, byl zde snímek s rozlišením pod uvedenou hranicí 400x400 pixelů a na jednom snímku se také projevovalo kolísání jasu žárovkového svítidla.

	Dekódování **proběhlo** (ks)	Dekódování **neproběhlo** (ks)	Úspěšnost (%)
Snímky **vyhovující** podmínkám	**6**	**0**	**100**
Snímky **nevyhovující** podmínkám	**1**	**5**	**16**

Tab. 8 – Zhodnocení testování programu

6. ZÁVĚR

Seznámil jsem se s nejznámějšími čárovými kódy používanými v logistice, obchodech a průmyslu, jejichž přehled je uveden v kapitole 2.1. Z těchto kódů jsem si vybral QR kód, kterému jsem se věnoval nejvíce. Nastudoval jsem jeho složení, důležité prvky a jeho princip. QR kód, je kód o mnoho složitější než například EAN kód. Používá spoustu prvků, které urychlují a usnadňují čtení kódu, například oproti Datametrix kódu.

Univerzální popis QR kódu je rozsáhlý dokument (více jak 100 stran) a značně by zvyšoval objem této práce, a proto jsem se snažil popsat kód tak, aby byl popis stručný, názorný a zároveň dostatečný. Pro zjednodušení jsem převážně používal jen QR kód velikosti 1 (verze 1). To znamená, že je 21 x 21 modulů velký. Na odlišnosti kódu jiných velikostí upozorňuji přímo v daných kapitolách.

Vytvořil jsem také vzorový příklad převodu dat do QR kódu krok po kroku. Ukázal jsem postup tak, aby byl srozumitelný a poukazoval i na odlišnosti postupu v jiných případech. Příklad byl doplněn obrázky, které charakterizují postupnou tvorbu kódu. Výsledek byl porovnán s kódem vytvořeným pomocí online generátoru (např. pomocí: [15]) a byl totožný.

Mírně odlišnou částí práce bylo pořízení databáze snímků (kapitola 3), ta se skládá jednak z QR kódů vytvořenými online generátory, ale hlavně fotoaparáty různých typů. Kódy byly vytištěny na matný klasický a lesklý fotografický papír. U lesklého papíru jsem pozoroval odrazy použitého přisvětlení, které poškozovaly kvalitu snímku. Záleželo na pozici světla a snímacího zařízení, kdy bylo zapotřebí najít nejvhodnější scénu, která odrazy eliminuje.

Jako nejlepší považuji scénu, kdy snímací zařízení je přímo nad kódem a světlo je umístěno mimo půdorys snímku. Dochází však k nerovnoměrnému osvětlení kódu, což by mohlo činit problémy při jeho zpracování. Tato nevýhoda by se však dala snadno vyřešit přidáním dalšího zdroje světla na protější stranu opět mimo půdorys snímku. Zdrojů by mohlo být použito i více tak, že by obklopovaly objektiv snímacího zařízení.

V další kapitole (4) jsem uvedl popis navrženého řešení. Tedy algoritmu, který zpracovává QR kód. Jeho vstupem je snímek pořízený snímacím zařízením a výstupem jsou data, která byla vložena do kódu při jeho tvorbě. Tento algoritmus má několik omezení. Předpokládá obraz QR kódu, který na své levé a horní straně má jen čistě bílou oblast. Kód by také neměl být pootočen o více jak 1° - 2°. Velikost sejmutého QR kódu na snímku by měla být alespoň 400 x 400 pixelů a obraz by neměl být geometricky deformován. Tato omezení by se dala eliminovat, bylo by však zapotřebí více času (například semestr – jako u diplomové práce) k úpravě některých modulů algoritmu.

QR kódy přinášejí obrovské množství variací velikostí, typů dat a dalších podrobností. Proto byl pro zachování přehlednosti zdrojového kódu algoritmu, omezen datový typ jen na alfanumerický. Při splnění těchto podmínek je algoritmus schopen dekódovat data uložená v QR kódu s přesností 100%, tak jak bylo uvedeno v kapitole 5.

7. SEZNAM POUŽITÝCH ZDROJŮ

[1] Šonka, M.; Hlaváč, V.: *Počítačové vidění*, Grada, Praha 1992, ISBN 80-85424-67-3

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

[3] Hlaváč, V.; Sedláček, M.: *Zpracování signálu a obrazu*, skriptum ČVUT 2001

[4] BRADSKI, Gary; KAEHLER, Adrian: *Learning OpenCV: Computer Vision with the OpenCV Library*, O'REILLY, 2008, ISBN 978-0-596-51613-0

[5] ISO/IEC 18004:2006. *Information technology - Automatic identification and data capture techniques - QR Code 2005 bar code symbology specification [draft]*

[6] MANDAU, Markus; KLEGA, Vratislav. QR Kódy: Sejmout a na web. CHIP: *Magazín informačních technologií*, květen 2008, č. 5, s. 32–33.

[7] Kodys: *Čárový kód* [online]. Cit. 26.11.2009. Dostupné z WWW: <http://www.kodys.cz/carovy-kod.html>

[8] Kodys: Čárový kód: UCC/EAN128 [online]. Cit. 29.11.2009. Dostupné z WWW: <http://www.kodys.cz/carovy-kod/ucc-ean-128.html>

[9] Kodys: *Čárový kód: PDF 417* [online]. Cit. 29.11.2009. Dostupné z WWW: <http://www.kodys.cz/carovy-kod/pdf-417.html>

[10] Kodys: *Čárový kód: CODE 39* [online]. Cit. 29.11.2009. Dostupné z WWW: <http://www.kodys.cz/carovy-kod/code-39.html>

[11] BELOS Trade s.r.o.: *Čárový kód* [online]. Cit. 26.11.2009. Dostupné z WWW: <http://www.carovykod.com/index.php?id=2&lang=cz>

[12] Denso wave: *QR code standardization* [online]. Cit. 26.11.2009. Dostupné z WWW: <http://www.denso-wave.com/qrcode/qrstandard-e.html>

[13] Wikipedie: *Čárový kód* [online]. Cit. 28.11.2009. Dostupné z WWW: <http://cs.wikipedia.org/wiki/čárový_kód>

[14] Barcode Library: *Barcode Industrial 2 of 5* [online]. Cit. 26.11.2009. Dostupné z WWW: <http://www.barcodelib.com/java_barcode/barcode_symbologies/code2of5.html>

[15] KAYWA: *Generátor QR kódů* [online]. Cit. 28.11.2009. Dostupné z WWW: <http://qrcode.kaywa.com/>

[16] Quido magazín: *Objevy - Čárový kód* [online]. Cit. 28.11.2009. Dostupné z WWW: <http://www.quido.cz/objevy/carovykod.htm>

8. SEZNAM POUŽITÝCH ZKRATEK A SYMBOLŮ

LSB - Least Significant Bit - bit s nejnižší váhou

MSB - Most Significant Bit - bit s nejvyšší váhou

CW - Codeword - v překladu kódové slovo, je to 8 bitová binární část kódu obsahující data

Finder pattern = kotvící obrazec - slouží v rychlejšímu lokalizování QR kódu ve scéně

Alignment pattern = zarovnávací obrazec - pomáhá k softwarové rekonstrukci deformovaného kódu

Timing patterns = zaměřovací obrazec - určují hustotu souřadnicové sítě QR kódu

Verze - verzí u QR kódu je myšlena jeho velikost, označení verze a popisy všech verzí jsou převzaty z [5]

Bod (=modul) - bodem je myšlen jeden čtvereček (bílý nebo černý) v QR symbolu, je to nejmenší popsatelná část symbolu

Quiet zone - je oblast kolem QR kódu, která nemá obsahovat žádný potisk

9. SEZNAM PŘÍLOH

Příloha 1 - CD s elektronickou verzí této práce a zdrojovým kódem