VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ČTEČKA QR KÓDŮ PRO ANDROID

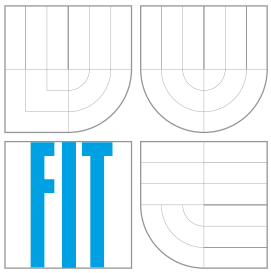
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

RADIM LOSKOT

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ČTEČKA QR KÓDŮ PRO ANDROID

QR READER FOR ANDROID

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — RADIM LOSKOT
AUTHOR

VEDOUCÍ PRÁCE — Ing. LUKÁŠ MARŠÍK
SUPERVISOR

BRNO 2012

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2011/2012

Zadání bakalářské práce

Řešitel: **Loskot Radim**
Obor: Informační technologie
Téma: **Čtečka QR kódů pro Android**
QR Reader for Android
Kategorie: Zpracování obrazu

Pokyny:

1. Seznamte se s základy zpracování obrazu a knihovnou OpenCV.
2. Osvojte si principy programování pro platformu Android a seznamte se s jejím API.
3. Prostudujte teorii QR kódů a vytipujte vhodný postup pro automatické rozpoznání.
4. Navrhněte aplikaci umožňující rozpoznání a přečtení QR kódu ve snímaném obraze.
5. Implementujte navrženou aplikaci pro mobilní platformu Android.
6. Proveďte sérii vhodných testů pro vyhodnocení výkonnosti a spolehlivosti použitého řešení.
7. Diskutujte budoucí rozšíření a modifikaci programu.

Literatura:

- G. Bradski, A. Kaehler: Learning OpenCV (Computer Vision with the OpenCV Library)
- Dále dle pokynů a doporučení vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- body 1 až 4

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Maršík Lukáš, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2011
Datum odevzdání: 16. května 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato bakalářská práce se věnuje problematice čtení dvourozměrných čárových QR kódů, návrhu a popisu implementace aplikace čtečky QR kódů pro mobilní platformu Android. V teoretické části analyzuje strukturu QR kódů včetně způsobu kódování obsažených informací a uvádí referenční přístupy, jakými lze tyto informace dekódovat. S ohledem na cílovou platformu Android navrhuje přístupy pro detekci QR kódů v reálném obraze a dekódování v zachyceném obraze z kamery mobilního zařízení. Závěrem hodnotí spolehlivost a výkonnost implementovaných funkcionalit a zamýšlí se nad možnostmi jejich dalšího budoucího rozšíření.

Abstract

This bachelor thesis focuses on the topic of reading the QR codes with the aim of designing and description of the implementation of QR codes' reader application for Android mobile platform. In theoretical part it analyzes structure of the QR codes including the way of information encoding and states the possible reference approaches of decoding them. With regard to the target platform Android it proposes algorithms for the detection of QR codes in the real time image and for decoding of the captured image from a mobile phone camera. At the end it evaluates the reliability and efficiency of the implemented functionalities and considers possibilities of their further development.

Klíčová slova

dvourozměrné čárové kódy, QR kódy, detekce, dekódování, čtečka QR kódů, Android, OpenCV

Keywords

2D barcodes, QR codes, detection, decoding, QR code reader, Android, OpenCV

Citace

Radim Loskot: Čtečka QR kódů pro Android, bakalářská práce, Brno, FIT VUT v Brně, 2012

Čtečka QR kódů pro Android

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Lukáše Maršíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Radim Loskot
14. května 2012

Poděkování

Na tomto místě bych rád poděkoval panu Ing. Lukáši Maršíkovi za vedení mé práce, věcné připomínky a poskytnutí mobilních zařízení k otestování finální aplikace.

Obsah

Kapitola 1

Úvod

Čárové kódy jistě není třeba dlouze představovat, setkáváme se s nimi téměř každý den při nakupování a markování zboží. Svojí schopností uchovávat informaci, která jednoznačně identifikuje daný typ zboží, společně s pokladními systémy usnadňují a zrychlují proces prodeje zboží většině současným prodejcům.

Čárové kódy jsou primárně určeny pro čtení strojem, mohou však obsahovat i textové části, které lze využít, pokud je kód poškozen či ho nelze z jiného důvodu přečíst. Pro čtení čárových kódů se využívá čteček čárových kódů. Existuje celá řada druhů čteček. Jednotlivé čtečky se odlišují ve schopnosti snímání kódů z jiných povrchů materiálu, za odlišných okolních světelných podmínek a v samotné podpoře čárových kódů, jež mohou číst. Při čtení čárových kódů se využívá odrazivosti světla, jehož zdrojem může být sama čtečka nebo okolní světlo. Pro zachycení světla se využívá řady (matice) CCD[1] snímačů nebo fotodiod, přičemž, v dnes již výjimečných případech, může být použito i pouze jedné fotodiody. CCD snímače nacházejí své uplatnění velmi často při čtení dvourozměrných čárových kódů, zatímco fotodiody při čtení klasických čárových kódů. [16]

Donedávna jsme se mohli s čárovými kódy většinou setkat pouze při prodeji, logistice a výrobě zboží. S příchodem mobilních telefonů s fotoaparáty, které zjednodušeně řečeno jsou složeny také z CCD snímačů, se naskytla zajímavá otázka využití fotoaparátu pro přenos dat. V té době již existovaly technologie infračerveného přenosu, Bluetooth aj. Přenos informací do mobilního telefonu z tištěné formy by byl ovšem novinkou. Přenos pomocí čárových kódů se přímo nabízel a o jeho prosazení se postaraly a stále starají především marketingové firmy, které tímto způsobem prezentují odkazy na své inzerenty. Samotným průkopníkem v této oblasti je Japonsko, kde se jako vhodný čárový kód pro přenos informací ustálil QR kód[2]. Ani v této době nejsou ovšem čtečky QR kódů v mobilních telefonech řešeny konstrukčně, proto je nutné je implementovat na úrovni softwarové vrstvy pomocí přídavných aplikací. Tvorbou takové aplikace se zabývá tato bakalářská práce.

Práce si klade především za cíl zanalyzovat aktuální standard QR kódů, navrhnout vhodnou automatickou lokalizaci QR kódů v obraze z kamery mobilního telefonu a metodiku, s níž by bylo možné QR kódy dekódovat. Na základě poznatků z analýzy problematiky QR kódů a jejich extrakce z obrazu se práce věnuje návrhu samotné aplikace čtečky QR kódů a její implementační realizaci pro cílovou mobilní architekturu Android.

Historii dvourozměrných čárových kódů, důvody vzniku QR kódů pro přenos informací a detailní strukturu QR kódů lze nalézt v kapitole 2. Stručný přehled stále poměrně nové

[1] Charge-coupled device – elektronická součástka určená pro snímání obrazové informace

[2] Quick Response kód – dvourozměrný čárový kód tzv. kód rychlé odezvy

mobilní architektury Android je zobrazen v kapitole 3. Kapitola 4 obsahuje návrh algoritmu pro detekování a dekódování QR kódů z obrazu a samotný návrh cílové aplikace. V kapitole 5 je již popsána výsledná implementace aplikace. Poslední kapitola 6 hodnotí úspěšnost čtení QR kódů z obrazu, zabývá se možnými rozšířeními a případným dalším vývojem aplikace.

Kapitola 2

QR kódy

QR kódy[1] jsou dvourozměrné čárové kódy vyvinuté společností Denso Wave Inc. v roce 1994. Společnost na tento druh čárových kód také vlastní patent, patentová práva ovšem nejsou vykonávána. Svá data kódují pomocí tmavých a světlých bodů a přenášejí je jak v horizontálním, tak i vertikálním směru. Z tohoto důvodu se pro jejich čtení využívá zejména matice CCD snímačů. Díky schopnosti kódování dat v obou směrech mohou přenést i více než stonásobně více dat než některé klasické čárové kódy, které kódují své informace pouze v jednom směru. [1]

QR kódy byly zprvu vyvinuté pro sledování vozidel během výrobního procesu. Svojí rychlou detekcí, dekódováním a velkým množstvím informací, které jsou schopny přenášet, se ale dostaly do širšího podvědomí a staly se populárními také pro přenos dat do mobilních telefonů. Bližší důvody jejich úspěchu, porovnání s jejich předchůdci a samotný proces standardizace lze nalézt v podkapitole 2.1.

Z hlediska schopnosti QR kódů přenášet velké množství informací na malé tištěné ploše, bylo při jejich návrhu nutné řešit vhodné prostředky pro přesnou lokalizaci a věrohodné čtení těchto kódů. Tyto prostředky jsou dopodrobna popsány v podkapitolách 2.2 a 2.3.

QR kódy mohou přenášet informace mnoha druhů, jejich interpretace v podobě dat jsou často aplikačně specifické. V mobilním průmyslu se ovšem během posledních let ustálilo pár zažitých formátů, které jsou dodržovány a podporovány většinou moderních čteček QR kódů. O jaké formáty se jedná, se můžeme dozvědět v podkapitole 2.4.

Poznatky o QR kódech v této kapitole, není-li citováno jinak, vychází z aktuální mezinárodní normy upravující formát QR kódů ISO/IEC 18004:2006 (dále jen „ISO norma“) [18].

2.1 Vývoj QR kódů

2.1.1 Přehled dvourozměrných kódů

Za rok vzniku dvourozměrných (2D) čárových kódů je považován rok 1984, kdy byl poprvé použit automobilovou společností AIAG tzv. skládaný čárový kód. Tento kód byl složen z celkem 4 klasických jednorozměrných (1D) čárových kódů značených jako Code 39, které kódovaly číslo dílů, množství, dodavatele a sériové číslo. Jak je patrné, jedním z důvodů pro vznik 2D kódů byla právě potřeba přenášet více informací (strukturovaných informací),

[1] Slovo „QR kód“ je registrovaná obchodní značka společnosti DENSO WAVE INCORPORATED v zemích Japonska, Spojených států Amerických, Austrálie a Evropy.

ale také možnost uchovávat data redundantně, což se ovšem odvíjí a vychází také z větší kapacity kódu. [19]

První kód, který byl představen jako ryze dvourozměrný, byl Code 49 (obrázek 2.1). Jednalo se opět o skládaný čárový kód, tzn. že byl složen z více jednorozměrných kódů. Tentokrát ovšem na rozdíl od svého předchůdce jednotlivé 1D kódy tvořily kompaktní celek, z toho důvodu byly jednotlivé řádky kódů odděleny tmavou dělící linií. Dalšími známými skládanými kódy jsou Code 16K, Codablock F, PDF 417, MicroPDF417 či ISS SuperCode. Některé z nich již nepotřebují oddělující řádky jako např. PDF 417 (obrázek 2.1), který konkrétně využívá shlukování (clusteringu) pro odlišení řádků vzájemně mezi sebou. Při shlukování je využito tří unikátních kódových sad (clusterů), které jsou každé tři řádky střídány. Aplikace rozdílných clusterů na vzájemně sousední řádky vždy zajistí vytvoření pomyslných oddělujících linií, které lze pak strojově zrekonstruovat. [14, 6]

Obrázek 2.1: Skládané dvourozměrné kódy – Code 49 (vlevo) a PDF417 (vpravo) [8]

Souběžně se se skládanými kódy vyvíjí i druhá skupina dvourozměrných kódů tzv. maticové kódy. Maticové kódy, kam mj. patří i QR kód, jsou tvořeny body, jejichž výška a šířka má pevný a stejný rozměr. Pro svůj stejný rozměr kódování v horizontálním i vertikálním směru nejsou vertikálně redundantní, tzn. je téměř povinností využití robustních samoopravných kódů. Naopak díky této vlastnosti dokáží uchovat více informací než skládané kódy. Kódy nemusí být čteny po řádcích, při jejich čtení se využívá čtecích mřížek. Představiteli maticových kódů jsou např. kódy Data Matrix, Aztec Code, MaxiCode aj. [29]

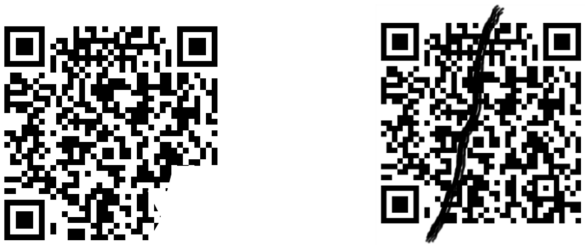
Obrázek 2.2: Maticové dvourozměrné kódy – Data Matrix, Aztec Code a MaxiCode [8]

V současné době již existuje přes 20 různých 2D kódů a stále se vyvíjejí další. Využívají se především pro značení součástí a výrobků – metoda DPM[2]. [29]

2.1.2 Porovnání QR kódů s ostatními 2D kódy

QR kódy lze od ostatních 2D kódů rozeznat podle výrazných značek ve třech rozích kódu. Tyto značky slouží pro rychlou detekci v rozmanitém okolním prostředí a upevnění čtecí mřížky nezávisle na natočení kódu. Mechanismus obnovy, který využívají (kapitola 2.3.3), dokáže obnovit nevalidní a poničené segmenty dat až do 30 % poškození (obrázek 2.3).

[2]Direct Part Marking – přímé značení součástí



Obrázek 2.3: Ukázka poškozených, ale opravitelných QR kódů

Porovnání s ostatními 2D kódy, zmíněnými v předešlé kapitole 2.1.1, lze vidět v následující tabulce 2.1.

	QR kód (maticový)	**PDF417** (skládaný)	**Data Matrix** (maticový)	**MaxiCode** (maticový)
Kapacita režimů				
Číselný	7089	2710	3116	138
Alfanumerický	4296	1850	2355	93
Binární	2953	1018	1556	-
Kanji	1817	554	778	-
Výhody	malé rozměry vysoká kapacita rychlost čtení	vysoká kapacita	malé rozměry	rychlost čtení

Tabulka 2.1: Porovnání dvourozměrných kódů [1]

2.1.3 Standardizace QR kódů

QR kód od své první aplikace v automobilovém průmyslu byl schválen mnoha standardizačními organizacemi včetně mezinárodních (tabulka 2.2). Abychom plně porozuměli aktuální mezinárodní normě ISO/IEC 18004:2006, která strukturu QR kódů upravuje, tak se v této kapitole podíváme na průběh standardizace, zejména té mezinárodní, trochu blíže.

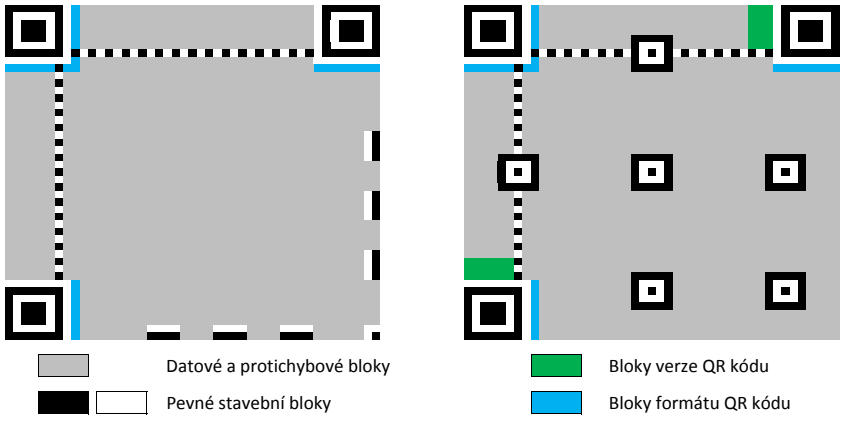
Datum	**Vydaný standard**
Říjen 1997	AIM standard (AIM-ITS 97/001)
Březen 1998	JEIDA standard (JEIDA-55)
Leden 1999	JIS standard (JIS X 0510)
Červen 2000	ISO standard (ISO/IEC 18004:2000)
Listopad 2004	2. vydání normy JIS X 0510
Září 2006	2. vydání normy ISO 18004 (ISO/IEC 18004:2006)

Tabulka 2.2: Standardizace QR kódů v letech [10]

První mezinárodní norma, která se QR kódy pokusila standardizovat, byla norma organizace AIM v roce 1997. Tato norma se snažila držet striktně formy QR kódů, které se

doposud používaly v aplikačně uzavřeném průmyslovém segmentu, skupinu QR kódů sjednotila a položila základy pro ostatní normy. Specifikováno bylo celkem 14 verzí QR kódů. Ve 20. století byl QR kód dále standardizován japonskými organizacemi JIS a JEIDA, a to zejména díky své schopnosti přenášet velice výhodně znaky japonských abeced.

V roce 2000 standardizovala QR kódy organizace ISO, přičemž její norma vychází z normy AIM, kterou ovšem již nedoporučuje dále používat v otevřené aplikační sféře. Starou normu AIM značně upravuje, přidává nové a odstraňuje staré stavební (konstrukční) prvky[3]. Specifikuje celkem 40 verzí QR kódů. Z důvodu kompatibility s normou AIM zavádí tzv. rodinu QR kódů, kde staré QR kódy označuje QR kódy modelu 1 a nové QR kódy modelu 2 (obrázek 2.4).

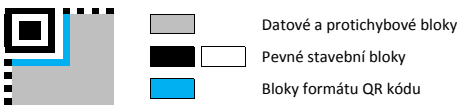


Obrázek 2.4: Porovnaní QR kódu stejné verze modelu 1 (vlevo) a modelu 2 (vpravo)

Aktuální norma ISO/IEC 18004:2006 již přímo rozšiřuje ISO normu vydanou v roce 2000. Nově přidává k rodině QR kódů další dva typy: QR kód 2005 a Micro QR kód.

QR kód 2005 vychází z QR kódu modelu 2, nepřidává žádné nové stavební prvky a ani nemění způsob kódování dat. Od QR kódu modelu 2 se QR kód 2005 odlišuje zejména v možnostech, jakými může být reprezentován. Byla přidána možnost zrcadlení kódu, jeho existence na tmavém povrchu s invertovanými body a možnost specifikace alternativní výchozí znakové sady zakódovaných dat v podobě textu. QR kódy modelu 2 jsou tedy plně kompatibilní s QR kódy 2005.

Kromě QR kódu 2005 byl v normě ISO/IEC 18004:2006 nově definován Micro QR kód, který byl už předtím standardizován v roce 2004 JIS organizací. Tento kód se od předešlých kódů liší výrazně – má velmi malé rozměry a dokáže přenést malé množství dat. Z důvodu malých rozměrů se používá často pro značení elektrických obvodů a součástek [7]. Aby ale přesto mohl o takto malých rozměrech pojmout co nejvíce dat, značně redukuje počet stavebních prvků. Norma definuje celkem 4 verze Micro QR kódů značené jako M1 až M4.

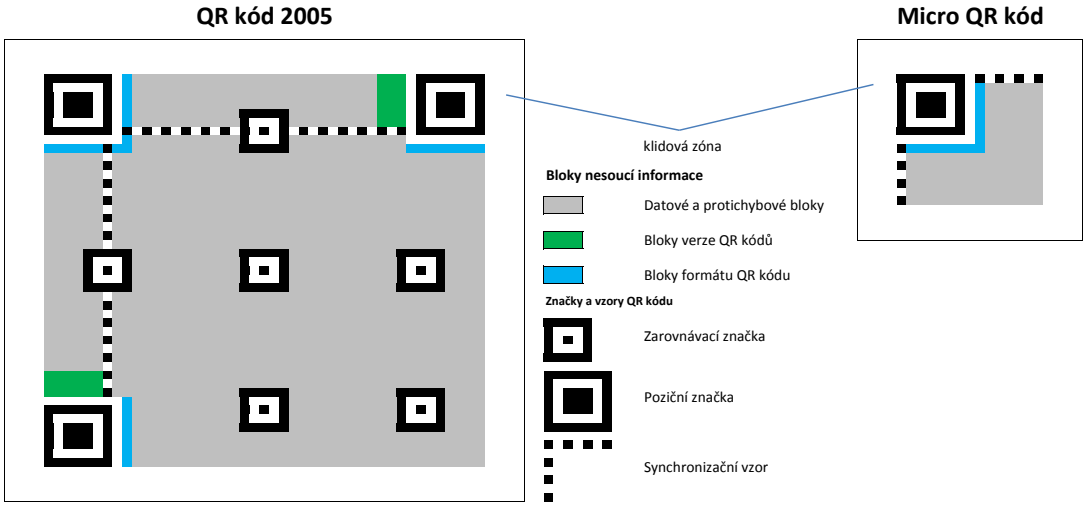


Obrázek 2.5: Micro QR kód

[3]stavební prvek – Souhrnné označení pro ustálené značky a vzory, které nenesou žádná data.

2.2 Struktura QR kódů

QR kód 2005 a Micro QR kód jsou složeny z pevných značek (vzorů), které musí obsahovat každý kód v dané verzi. Pevné značky jsou umístěny do kódu z důvodu rychlé lokalizace kódu, rychlého a věrohodnějšího čtení. Počet pevných značek se v závislosti na verzi kódu může měnit. Každý QR kód obsahuje také minimálně metainformaci o formátu zakódovaných dat. QR kód 2005 pak od verze 7 a výše musí obsahovat i metainformaci o své verzi. Každý kód musí být ohraničen klidovou zónou, která je alespoň široká jako 4 body kódu.



Obrázek 2.6: Struktura QR kódu 2005 (znázorněna verze 7)

Nejdůležitějšími značkami QR kódů jsou tzv. poziční (lokalizační) značky, které slouží pro rychlou lokalizaci kódu v obraze. Rozložení tmavých a světlých bodů poziční značky je vertikálně a horizontálně totožné, tmavé body se střídají se světlými s poměrem 1:1:3:1:1. Každá poziční značka musí byt oddělena od zóny dat pásem světlých bodů. QR kód 2005 obsahuje celkem tři poziční značky, Micro QR pouze jednu.

Jelikož kód může být zobrazen v obraze v perspektivní projekci, ale i zdeformován zakřiveným povrchem, na kterém je vytištěn, obsahuje kód i zarovnávací značky a synchronizační vzory. Tyto prvky slouží pro snadnější transformaci kódu z perspektivního či zakřiveného zobrazení do normalizovaného zobrazení pro dekódování kódu. Zarovnávací značky jsou umístěny v kódu na přesných pozicích, tak jak je definováno v příloze E ISO normy. Micro QR kód a QR kód 2005 verze 1 neobsahuje žádné zarovnávací značky. Synchronizační vzory existují dva, jeden horizontální, druhý vertikální. Oba začínají a končí na pevně daných pozicích, které jsou totožné v každé verzi QR kódu 2005 a nejinak tomu je i u Micro QR kódu. Své uplatnění nachází při odměřování bodů uvnitř datové sekce, jejichž polohy lze získat podle středů jednotlivých bodů synchronizačních vzorů.

Z hlediska rozměrů má QR kód 2005 verze 1 rozměry 21x21 bodů. Každá další verze rozšiřuje kód předcházející verze o 4 body na výšku i šířku. Z toho plyne, že verze 40 má rozměry 177x177 bodů. Nejmenší verze Micro QR kódu (M1) začíná na rozměrech 11x11 bodů. S každou další verzí se rozměry Micro QR kódu postupně navyšují o dva body, verze M4 tedy má 17x17 bodů.

2.3 Fyzická vrstva

V předcházející kapitole jsme popsali strukturu QR kódů, zmínili jsme se i o blocích metainformací a datových blocích. V této kapitole se budeme zabývat právě formátem a způsobem kódování informací, které jsou obsažené v těchto blocích, ale také i samotným dekódováním, které bude poté využito při tvorbě čtečky QR kódů.

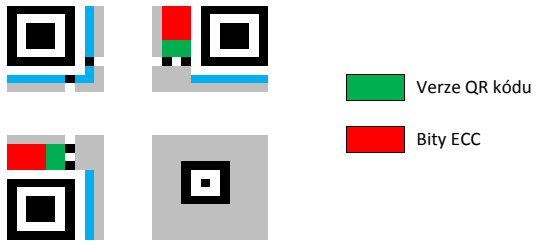
2.3.1 Formát a kódování metainformací

Samotná informační data metainformací nejsou kódována žádným kódem, nicméně každý blok s metainformacemi obsahuje také bity kontroly a korekce chyby a právě pro jejich výpočet je zapotřebí využít kódování BCH – Bose-Chaudhuri-Hocquenghem (příloha B).

Z důvodu malého počtu možných kombinací, které mohou BCH kódováním vzniknout, a náročnosti výpočtu kódu, je vhodné si pro kódování i dekódování metainformací vytvořit vyhledávací tabulku. Pro dekódování norma použití vyhledávací tabulky přímo doporučuje. Jelikož je minimální Hammingova vzdálenost pro použité samoopravné BCH kódy v obou metainformacích rovna třem, lze během čtení metainformací opravit maximálně tři bitové chyby. V případě využití vyhledávací tabulky nám tedy stačí nalézt záznam, který se shoduje v nejvíce bitech a přitom se liší maximálně ve třech bitech.

Formát metainformace verze QR kódu

Přikládat do QR kódu blok s metainformací o verzi kódu požaduje norma pouze pro QR kódy 2005 od verze 7 a výše. Jedinou informací, kterou tento blok nese je informace o verzi QR kódu, která je v něm uložena jako číslo na šesti bitech. Zbylé bity bloku jsou bity korekce chyby (ECC), které jsou k samotnému číslu na závěr připojeny. Metainformace verze QR kódu je uložena vedle dvou pozičních značek redundantně.



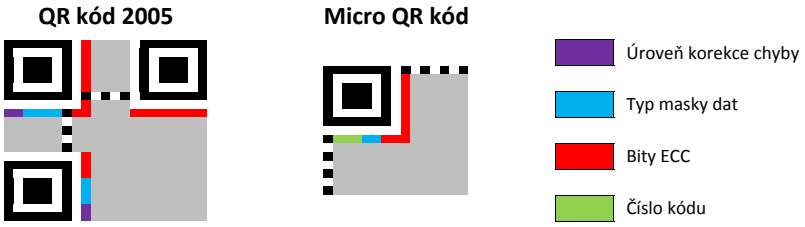
Obrázek 2.7: Znázornění bloků s metainformací verze QR kódu

Formát metainformace formátu dat

Každý QR kód, již od modelu 1, má blok s metainformací o způsobu zakódování výsledného vzoru dat.

V případě QR kódu 2005 tento blok obsahuje informaci o úrovni použité korekce chyb (2 bity), kterou kód disponuje (kapitola 2.3.3 tab. 2.3). Druhým údajem, který v bloku můžeme nalézt, je informace o masce (3 bity), která byla na výsledný vzor s daty aplikována za účelem odstranění nežádoucích vzorů z dat podobajících se pozičním značkám (kapitola 2.3.4 obrázek 2.13).

Micro QR kód blok metainformace také využívá pro uložení informace o masce (pouze 2 bity), ale jako druhou informaci přenáší číslo kódu (3 bity), které v sobě zapouzdřuje úroveň použité korekce chyby a samotnou verzi Micro QR kódu.



Obrázek 2.8: Znázornění bloků s metainformací formátu QR kódu

2.3.2 Kódování dat

Datový proud, jenž má být QR kódem přenášen, by neměl být vložen do kódu přímo, nýbrž zapouzdřen do jednoho či více datových segmentů. Datové segmenty přitom mohou být řazeny v závěsu za sebe. Každý segment musí v hlavičce obsahovat své unikátní označení tzv. indikátor režimu. Segmentování v čárových kódech slouží k efektivnějšímu přenosu specifických typů dat, jako jsou řetězce čísel, alfanumerických či znaků japonských abeced. Zatímco ve znakovém režimu bychom mohli přenést na 3 bytech pouze 3 cifry, v numerickém režimu jich můžeme přenést celkem 9. Celá sekvence segmentů musí být zakončena speciálním segmentem tzv. terminátorem.

SEGMENT 1			SEGMENT 2			...	SEGMENT N			TERMINÁTOR
Indikátor režimu	Délka datového proudu	Datový proud	Indikátor režimu	Délka datového proudu	Datový proud	...	Indikátor režimu	Délka datového proudu	Datový proud	

Obrázek 2.9: Ukázka kódování dat do datových segmentů

Pro QR kódy existují následující režimy (typy segmentů):

- **numerický** – slouží pro přenos čísel, číslice se snaží kódovat do skupin o třech cifrách a přenášet je na 10 bitech. Obsahuje-li skupina méně než 3 cifry, poté jsou přenášeny tyto cifry na 7 nebo 4 bitech;
- **alfanumerický** – přenáší alfanumerické znaky a některé další symboly – celkem 45 znaků;
- **binární** – nespecifikuje datům žádnou sémantiku a přenáší je po bytech;
- **Kanji** – efektivně kóduje znaky japonských abeced;
- **ECI** – režim neslouží pro přenos dat, nýbrž pro specifikaci kódové sady znaků v následujících segmentech. Výchozí sadou je ISO/IEC 8859-1;
- **FNC1** – FNC1 režim nepřenáší data, pouze značí, že následovaná data podléhají specifickému kódování EAN.UCC nebo jinému registrovanému AIM organizací;

- **strukturovaný** – pokud je segment s tímto režimem obsažen v QR kódu, říká nám, že data v jeho závěsu jsou neúplná a rozdělena do více QR kódů. Pro sestavení a uspořádání kompletních dat poté slouží sekvenční číslo, které je zahrnuto v každém strukturovaném segmentu.

2.3.3 Metoda zotavení z chyb Reed-Solomon

Pro zotavení dat z chyb je v QR kódech využívána bloková a nebinární Reed-Solomon (RS) kódovací metoda. Mimo QR kódů se s ní můžeme setkat např. u CD a DVD nosičů, ADSL, mobilních telefonů, mezisatelitní komunikace či televizního přenosu DVB. Reed-Solomon metoda má podklad v diskrétní matematice a kódy, které generuje, se dají vyjádřit cyklickým posunem a lineární kombinací jiného kódového slova. Jedná se tedy o lineární a cyklický kód.

Reed-Solomonovy kódy (RS kódy) vznikly specializací BCH kódů. Jejich sestavení se uskutečňuje pomocí operací využívajících prvků Galoisova tělesa $GF(2^m)$. Jelikož se jedná o nebinární kódování, pracuje RS kódovací metoda pouze se symboly (nikoliv bity), jež vstupují do dekódovacího procesu a mohou být také opraveny. Těleso $GF(2^m)$ je poté rozšířením binárního tělesa $GF(2)$ a ke dvěma prvkům abecedy $\{0, 1\}$ přidává další prvky vyjádřené jako mocniny primitivního prvku α (pro QR kódy $\alpha = 2$):

$$F = \{0, 1, \alpha^1, \alpha^2, \ldots, \alpha^{2^m-4}, \alpha^{2^m-3}, \alpha^{2^m-2}\}. \tag{2.1}$$

V případě QR kódů obsahuje vždy Galoisovo těleso 2^8 prvků, neboť kódové symboly QR kódu jsou 8 bitů dlouhé. [26]

Výsledný RS kód je tvořen generujícími polynomy $g(x)$ stupně $n - k$, kde n je délka výsledného kódu a k délka nezakódovaných informačních dat. Vždy přitom musí platit, že generující polynom dělí polynom $x^n - 1$ beze zbytku. Pro QR kódy jsou již v závislosti na k i n generující polynomy sestaveny (viz ISO norma příloha A).

Pokud uvážíme informační data v polynomiálním tvaru

$$f(x) = f_0 + f_1 x + f_2 x^2 + \ldots + f_{n-1} x^{n-1}, \tag{2.2}$$

výsledná paritní část informačních dat, jež je k datům na závěr připojena, je zjistitelná vztahem

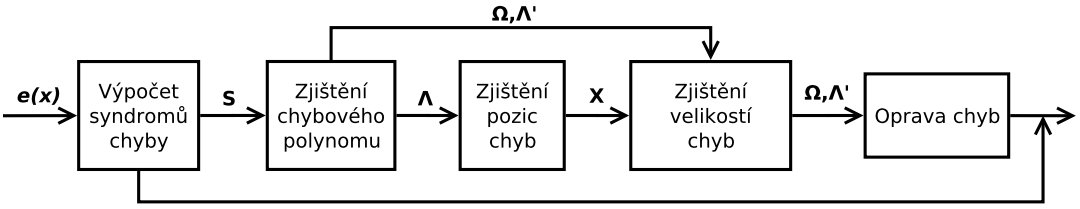
$$\frac{f(x)x^{n-k}}{g(x)} = h(x) + \frac{r(x)}{g(x)} \tag{2.3}$$

jako zbytek po dělení $r(x)$. [24]

V praxi se pro generování kódu vyžívá generující neboli bázové matice $\mathbf{G}(\mathbf{x})$ (rovnice 2.4), která vychází z cyklických a lineárních vlastností RS kódu. Matice má celkem k lineárně nezávislých řádků, které jsou tvořeny cyklicky posunutými kódovými slovy. Výsledný kód je poté zjištěn pomocí lineární kombinace jednotlivých řádků matice. [23]

$$\mathbf{G} = \begin{pmatrix} g_{n-k} & g_{n-k-1} & \cdots & \cdots & \cdots & \cdots & g_1 & g_0 & 0 & 0 & \ldots & 0 \\ 0 & g_{n-k} & g_{n-k-1} & \cdots & \cdots & \cdots & \cdots & g_1 & g_0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & g_{n-k} & g_{n-k-1} & \cdots & \cdots & \cdots & \cdots & g_1 & g_0 \end{pmatrix} \tag{2.4}$$

Proces dekódování a případné opravy chyby v RS kódu je více komplikovaný. Jeho matematický popis by překračoval rozsah této práce. Základní podstata procesu je však vystižena v diagramu na obrázku 2.10. Pro určení chybového polynomu existuje více metod, z těch efektivnějších můžeme jmenovat Euklidovský nebo Berlekampův algoritmus. Během řešení kořenů chybového mnohočlenu se může uplatnit algoritmus Chienova vyhledávání a při zjišťování velikosti chyby Forneyův algoritmus. [24]



Obrázek 2.10: Proces dekódování Reed Solomon kódu

Podmínka opravitelnosti RS kódu, kde t je počet detekovaných chyb, je dána: [26]

$$t \leq \frac{n-k}{2}. \tag{2.5}$$

Data v QR kódech mohou být zabezpečena čtyřmi různými úrovněmi L, M, Q a H (tabulka 2.3). Z pohledu RS kódů změna úrovně zabezpečení znamená zvolení jiné délky n pro výslednou kódovou zprávu, přičemž délka zprávy nesmí přesáhnout $2^m + 1$ symbolů, neboť pro RS kódy platí: [26]

$$0 < k < n < 2^m + 2. \tag{2.6}$$

Aby byla předchozí podmínka splněna, jsou pro větší QR kódy datové segmenty rozděleny do bloků a zabezpečení probíhá nad více menšími vzájemně nezávislými bloky (viz ISO norma tabulka 9)

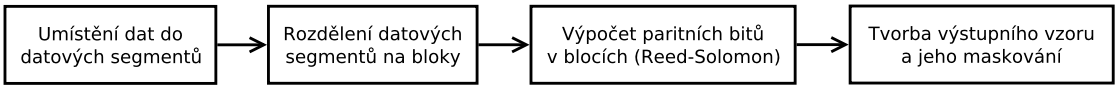
Úroveň korekce chyb	**Možnost zotavení z chyb (cca %)**
L	7
M	15
Q	25
H	30

Tabulka 2.3: Možnosti zotavení dat v procentech

Aktuální aplikovaná úroveň zabezpečení pro daný kód je uložena v metainformaci formátu QR kódu. Zatímco QR kódy 2005 mohou pro svůj přenos využít všechny 4 úrovně, u Micro QR kódu je schopnost zabezpečení přímo závislá na verzi kódu. Pro verzi M1 např. neexistuje žádné zotavení z chyby, pouze detekce chyby, a vyjma verze M4, nepodporují Micro QR kódy korekci úrovně Q.

2.3.4 Tvorba výstupního vzoru

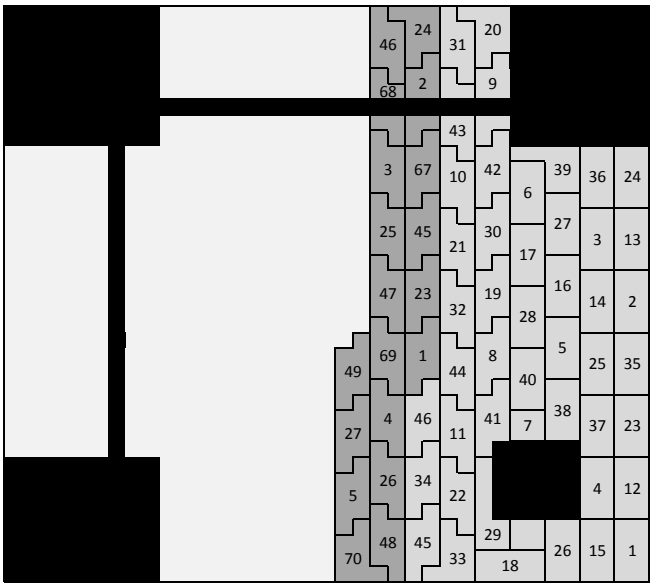
Pokud máme data vložena do datových segmentů s vhodně zvolenými režimy pro přenos (kapitola 2.3.2), datové segmenty rozděleny na bloky a vypočítány pro ně paritní bity korekce (kapitola 2.3.3), můžeme přistoupit k finálnímu umístění bloků do QR kódu.



Obrázek 2.11: Kroky vedoucí ke tvorbě výstupního vzoru

Kódová slova bloků jsou umísťována do QR kódu vertikálně v šíři 2 bodů od pravého dolního rohu směrem do levého horního rohu, přičemž po dosažení horní či dolní strany je směr zápisu zrcadlen. Kódová slova korekce jsou vždy z bloků vytržena a umísťována do kódu až po zápisu všech kódových slov dat. Pakliže musíme umístit více než jeden blok, je zapotřebí uplatnit prokládání kódových slov mezi bloky. Velikost prokládání je rovna počtu všech bloků a slouží pro rovnoměrné rozložení případných chyb do více nezávisle zabezpečených bloků.

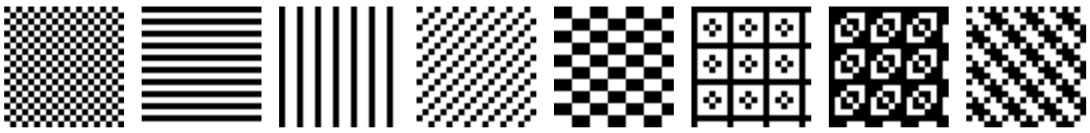
Pokud bychom uvážili QR kód verze 5 s úrovní korekce H, tento kód má celkem 4 zabezpečené bloky, kde první dva bloky obsahují 11 kódových slov dat a 22 kódových slov korekce, zbylé dva bloky obsahují akorát o 1 kódové slovo dat více, byla by kódová slova vkládána do QR kódu tak, jak lze vidět na následujícím obrázku 2.12.



Obrázek 2.12: Znázornění způsobu organizace kódových slov v QR kódu

Po umístění kódových slov do kódu nám zbývá pouze zvolit vhodnou masku pro případné odstranění lokalizačních značek (vzorů) ze vzoru s daty. Pro QR kód 2005 je definováno celkem 8 masek a pro Micro QR kód 4 masky. Vyhodnocení nejvhodnější masky probíhá otestováním všech masek nad vzorem s daty a vybrání té s nejlepšími vlastnostmi.

Maskování je založeno na aplikaci operace XOR nad jednotlivými body QR kódu a body masky.



Obrázek 2.13: Sada datových masek QR kódu

2.4 Aplikační vrstva

Aplikační vrstva pro import informací do mobilních telefonů přenášenými QR kódy není nijak standardizována a může být vždy specifická pro danou aplikaci. Postupem času se ustálily především dva univerzální formáty, které si zde zmíníme.

Prvním způsobem, jak přenášet informace, je kódovat je do URI [4, 22]. Jednotlivé typy informací jsou poté rozlišeny podle schémat. Samotné informace jsou obvykle přenášeny pomocí hierarchické části a dotazů:

```
schéma: hierarchická část [?dotaz] [#fragment]
```

Používaná URI schémata, jež jsou registrována organizací IANA, lze vidět v tabulce 2.4. Vyjma těchto schémat se můžeme setkat ale i s neregistrovanými schématy `URL`, `SMSTO` aj.

Přenos	**Norma**	**Formát přenášených dat**
SMS	RFC 5724	`sms:+15105550101?body=hello%20there`
webové stránky	RFC 2616	`http://www.example.com`
e-mailu	RFC 6068	`mailto:chris@example.com`
polohy	RFC 5870	`geo:13.4125,103.8667`
tel. čísla	RFC 3966	`tel:+19005550191`

Tabulka 2.4: Používaná a registrovaná schémata organizací IANA

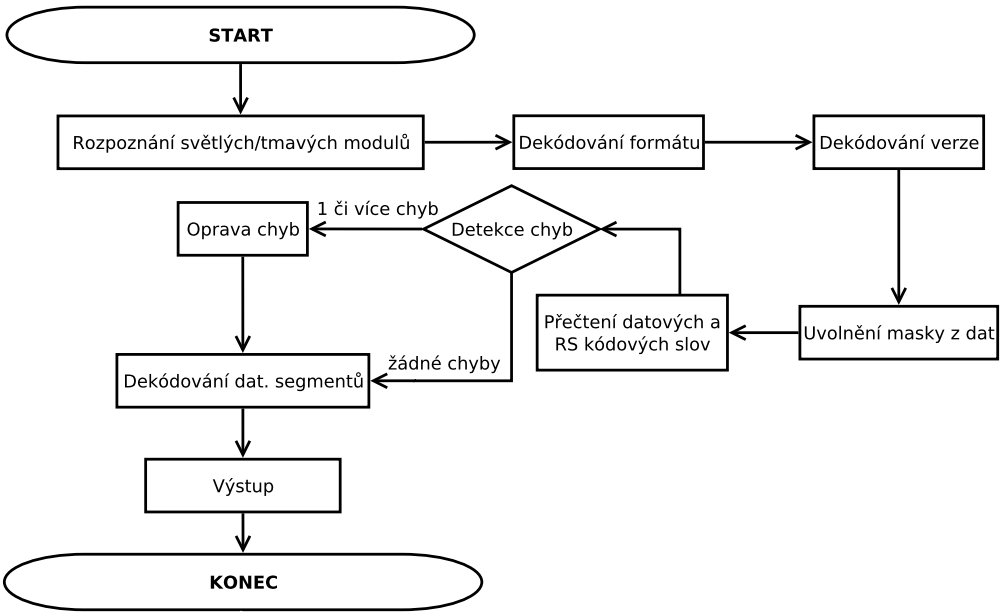
Druhým způsobem pro přenos informací je využití metodiky firmy NTT DoCoMo [15], která definuje formáty zejména pro vytváření záložek URL adres, přenos kontaktů a elektronických zpráv (tabulka 2.5). Pro přenos kontaktu se můžeme setkat i s formátem vCard (viz RFC 6350).

Přenos	**Formát přenášených dat**
webové záložky	`MEBKM:URL:example.com;TITLE:Example;;`
e-mailu	`MATMSG:TO:nick@example.org;SUB:subject;BODY:body;;`
kontaktu	`MECARD:TEL:+123456789;N:John Black;URL:example.com;;`

Tabulka 2.5: Příklady využití formátu firmy NTT DoCoMo

2.5 Čtení QR kódů

Pro čtení QR kódů aktuální ISO norma definuje referenční algoritmus, který je možné použít, ovšem není nutné, při tvorbě čtečky QR kódů. Základní kroky algoritmu, které lze považovat za obecné a uplatní se při tvorbě všech čteček QR kódů, můžeme vidět ve vývojovém diagramu na obrázku 2.14.



Obrázek 2.14: Referenční vývojový diagram znázorňující kroky čtení QR kódu

Jednotlivé kroky předešlého vývojového diagramu jsou ve znění referenčního algoritmu upřesněny. My si je zde ale nebudeme popisovat a pokusíme se nalézt v návrhu čtečky QR kódu vlastní, příp. modifikovaný algoritmus.

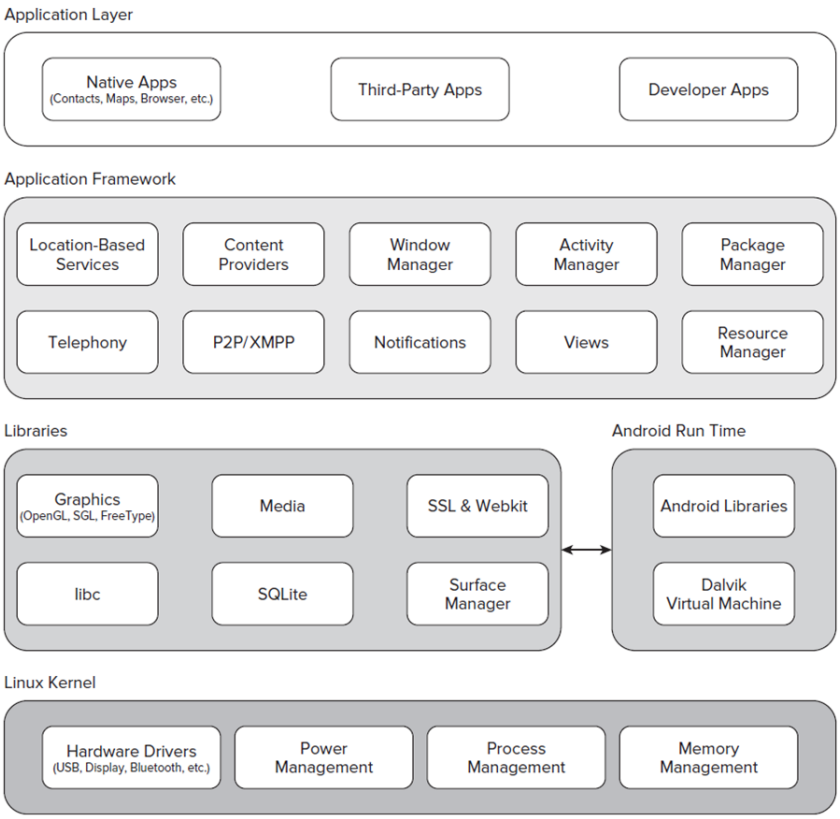
Kapitola 3

Google Android

Android je open-source softwarová platforma vyvíjená zejména pro mobilní zařízení. Zahrnuje v sobě operační systém, podpůrný systém pro běh aplikací a některé klíčové aplikace.

3.1 Architektura

Tato podkapitola čerpá z [17]. Základní architektura platformy Android se skládá z aplikační vrstvy, aplikačního frameworku, knihoven, Androidu runtime a Linuxového jádra. Přehledně ji můžeme vidět na následujícím obrázku 3.1.



Obrázek 3.1: Architektura platformy Android [25]

Aplikační vrstva, nejvyšší vrstva abstrakce, tvoří prostor pro běh aplikací. Zařízení používající Android jsou zpravidla dodávána již se sadou předinstalovaných optimalizovaných aplikací. Bývají jimi např. webový prohlížeč, aplikace na zpracování obrazu z kamery, SMS a e-mailový klient aj. Všechny aplikace v této vrstvě jsou napsány v jazyce Java a jsou spouštěny na speciálním virtuálním stroji zvaném Dalvik.

Aplikační framework nabízí vývojové prostředky pro tvorbu aplikací v aplikační vrstvě. API frameworku poskytuje plný přístup ke stejným komponentám, které používají klíčové aplikace systému. Architektura tvorby aplikací je založena na opětovném využívání komponent, které mohou být každou aplikací zveřejněny. Obdobný systém umožňuje i komponenty nahrazovat.

C/C++ knihovny poskytují zázemí pro běh některých klíčových API komponent, ale také samotného virtuálního stroje, proto musí být vždy součástí distribuce platformy. Hlavními knihovnami jsou: standardní systémová knihovna C (libc), knihovny pro práci s médii a grafikou (SGL, OpenGL, FreeType), relační databázový systém (SQLite) a další.

Android runtime obsahuje sadu klíčových knihoven, zajišťujících většinu funkcionality pro interpretaci jazyka Java, a samotný virtuální stroj Dalvik. Každá třída, která má být spuštěna pomocí virtuálního stroje, musí být převedena do formátu Dalvik Executable (.dex) pomocí nástroje `dx`, jež je přístupný v Android SDK. Aplikace jsou vždy spouštěny jako nový proces s vlastní instancí virtuálního stroje.

Linuxové jádro platformy Android je verze 6 a zprostředkovává služby zabezpečení, správu paměti, správu procesů, ovladačů aj.

3.2 Aplikace

Tato podkapitola čerpá z [3]. Aplikace pro Android jsou psány v jazyce Java, přičemž pro spuštění musí být zkompilovány do formátu Dalvik Executable. Pro distribuci by měly být aplikace zabaleny do balíku s příponou `.apk` a podepsány certifikátem. Při podepisování se využívá asymetrického šifrování, vlastníkem privátního klíče je sám autor aplikace. Nové aplikace je zapotřebí vždy nainstalovat. Během níž se každé aplikaci přiřadí vlastní uživatelský účet, s nímž bude spouštěna, vytvoří privátní složka pro ukládání souborů a balík s kompilovaným kódem aplikace překopíruje do systémových složek Androidu.

Základními stavebními bloky pro tvorbu aplikací na platformě Android jsou tzv. komponenty aplikace. O jejich existenci informujeme systém během instalace ve speciálním XML souboru (`AndroidManifest.xml`), který je nutno umístit do `.apk` archivu. Komponenty tvoří jakési přístupové a spouštěcí body, jejich kód je systémem vyvoláván na základě nějaké vnější události. Aktuálně existují celkem 4 druhy komponent: `Activity`, `Content Provider`, `Service` a `Broadcast receiver`.

Activity komponenta představuje jednoduchou obrazovku, na které lze vytvářet aplikace s uživatelským rozhraním. Tuto komponentu lze vyvolat metodou `startActivity()`. Pokud se dostane do popředí jiná Activity, aktuálně spuštěná Activity je pozastavena a může být uvolněna i z paměti a její stav uložen.

Service komponenta neposkytuje uživatelské rozhraní, běží jako služba na pozadí systému bez vědomí uživatele a lze ji spustit metodou `startService()`.

Broadcast receiver komponentou můžeme zachytávat vyslaná veřejná oznámení jinými komponentami. Komponenta, která vysílá veřejné oznámení, volá `sendBroadcast()`.

Content provider je speciální komponenta, která dokáže zprostředkovávat soukromá data aplikace jiným aplikacím.

Invokace komponent (mimo `Content provider`) probíhá pomocí instancí `Intent` objektů. Komponenta může být zavolána buď přímo, pokud známe třídu, která komponentu implementuje, nebo nepřímo požadavkem, jež lze předat `Intent` instancí. Jaké druhy požadavků může komponenta zpracovat, lze oznámit systému opět manifest souborem elementem `<intent-filter>`.

Kapitola 4

Návrh

V této kapitole budeme nejprve diskutovat požadavky na cílovou aplikaci a stanovíme si cíle, jichž budeme chtít dosáhnout. Dále si popíšeme, jakými vhodnými způsoby lze lokalizovat QR kód v obraze a přečíst jeho obsah. Aktuální ISO norma sice uvádí referenční postup, jak QR kódy číst, ale nespecifikuje konkrétní algoritmy a metody zpracování obrazu, jakými toho docílit. Z tohoto důvodu se pokusíme v této kapitole tyto metody navrhnout a upřesnit. Závěrem kapitoly bude prezentován základní návrh architektury aplikace.

4.1 Stanovení cílů

Pro návrh algoritmů detekce a čtení je zapotřebí zohlednit aplikační doménu, ve které budou pracovat, a podporované QR kódy, které budou schopny zpracovat.

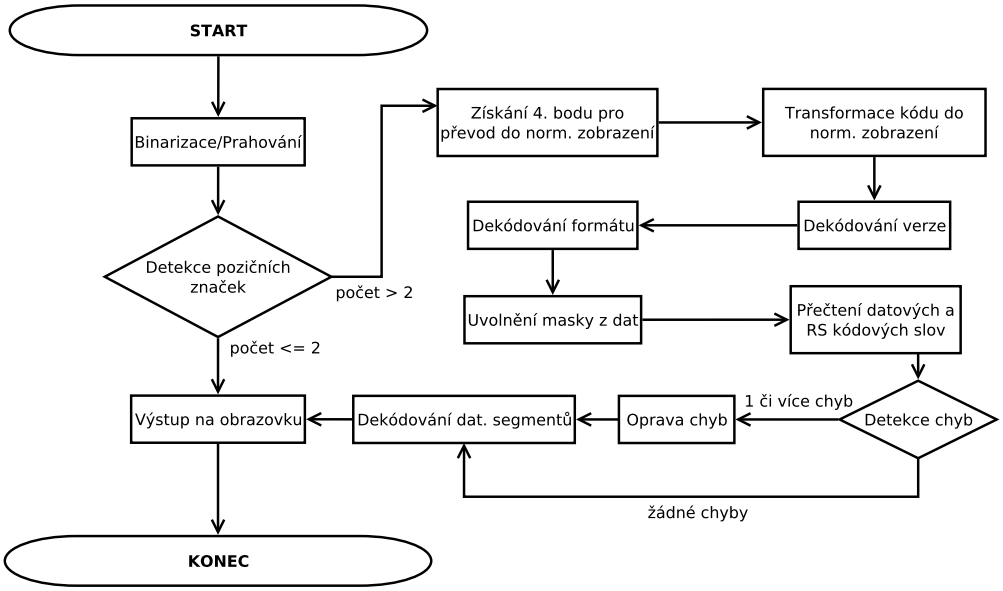
Víme, že aplikace bude tvořena pro čtení QR kódu mobilními přístroji, QR kód se tedy bude nacházet v prostoru a v perspektivním zobrazení. V rámci přípravy obrazu pro čtení bude nutno provést transformaci do normalizovaného zobrazení (kapitola 4.4, 4.5, 4.6). Snímání obrazu mobilními přístroji nám také většinou nezaručí rovnoměrně osvětlenou scénu, intenzita pro tmavý modul v jedné části QR kódu může odpovídat intenzitě pro světlý modul v jiné části kódu. Z čehož plyne důležitost volby vhodné metody převodu obrazu do dvouhodnotového (binárního) obrazu (kapitola 4.3).

Ačkoliv aktuální ISO norma definuje QR kódy 2005 a Micro QR kódy, v praxi se s nimi nikdy nesetkáme, alespoň co se mobilního průmyslu týče. Proto během návrhu a implementace budeme vycházet z běžně používaných QR kódů modelu 2.

Většina dnešních čteček umožnuje QR kódy detekovat a dekódovat v reálném obraze, což má mnoho výhod, jmenovitě např. schopnost eliminovat špatně dekódovaný QR kód a neinformovat o tomto neúspěchu uživatele. Naopak zpracování reálného obrazu je prováděno z výpočetních a hardwarových důvodů nad obrazem s nízkým rozlišením, což vede k problémům během zpracování větších QR kódů. Abychom umožnili práci i s kódy vyšších verzí, ale i použití automatického zaostření při snímání QR kódu, nebudeme QR kódy číst z reálného obrazu, nýbrž po jejich vyfocení. Nicméně bylo by vhodné, aby cílová čtečka informovala uživatele o skutečnosti, že QR kód byl v reálném obraze detekován a že lze předpokládat jeho úspěšné přečtení.

4.2 Čtení QR kódu

Základní navržený vývojový diagram, jak lze ke čtení QR kódu modelu 2 přistoupit, je zobrazen na obrázku 4.1.



Obrázek 4.1: Vývojový diagram navrženého postupu pro čtení QR kódů

Kroky diagramu pro dekódování informací a opravy chyby vychází z referenčního diagramu ISO normy (kapitola 2.5) a jsou doplněny o kroky binarizace, detekce pozičních značek a transformace QR kódu z perspektivního zobrazení, jež norma nerozvádí a zahrnuje je v jednom kroku. Z diagramu lze vyčíst, že dekódování je proveditelné pouze tehdy, jsou-li detekovány alespoň 3 poziční značky.

4.3 Binarizace obrazu

Příprava obrazu pro strojové čtení bývá často tím nejdůležitějším úkolem pro dosažení uspokojivých výsledků. Pro čtení QR kódů lze uplatnit tzv. binarizaci obrazu, kde se vícekanálový snímek převede na snímek ve stupni šedi, který je poté prahován. Prahováním vzniká obraz, který obsahuje pouze dvě hodnoty, jež představují tmavé a světlé pixely QR kódu.

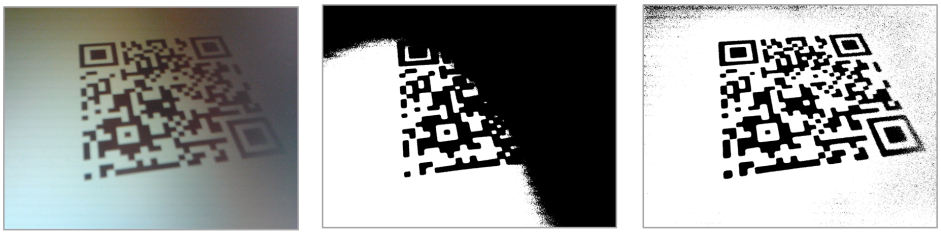
Pro převod barevného obrazu na obraz ve stupních šedi se obvykle využívá empirického vztahu barvy a intenzity

$$I = 0,299R + 0,587G + 0,114B, \tag{4.1}$$

kde R, G a B značí jednotlivé barevné složky.

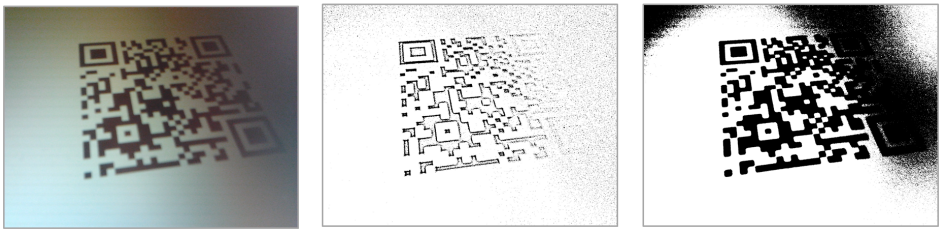
Zvolení správné metody a parametrů pro prahování obrazu bývá již složitějším úkolem a odvíjí se často na konkrétní doméně použití dané aplikace. Pro prahování obrazu existují zejména dva zásadní přístupy, kdy práh je volen globálně pro celý obraz (globální prahování) nebo lokálně (adaptivní prahování). Referenční algoritmus uvedený v ISO normě

pracuje s globálním prahováním. Přesto ale s ohledem na naši aplikaci, kde bude obraz pořizován z kamery za předem neznámých světelných podmínek, můžeme tvrdit, že použití adaptivního prahování bude výhodnější. Zvolení globální metody prahování by mohlo vést k negativním výsledkům (obrázek 4.2) a dalo by se zvážit např. při skenování QR kódu se zdrojem světla, pokud ovšem zanedbáme možnost výskytu přesvětlených míst.



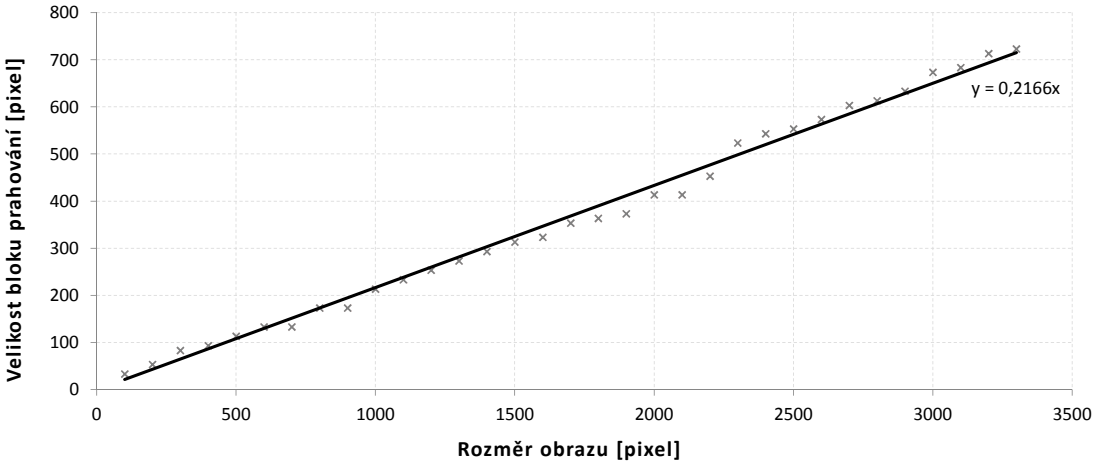
Obrázek 4.2: Porovnání globálního (uprostřed) a adaptivního prahování (vpravo)

K docílení adaptivní prahování se využívá většinou přístupu Chow a Kaneko nebo lokálního prahování. Z hlediska vysoké výpočetní složitosti prvního přístupu pro zpracování reálného obrazu, bude v aplikaci použito lokálního prahování. Tento přístup je založen na tvorbě matice, která se naplní vypočítanými prahy pro každý pixel vstupního obrazu. Hodnota prahu pro daný pixel vstupního obrazu pak může být získána jako aritmetický průměr, medián aj. jeho okolních pixelů, kde velikost okolí je specifikována velikostí bloku prahování. V případě zvolení malé velikosti bloku prahování může docházet v QR kódu ke tvorbě světlých míst, adaptivní prahování zde plní spíše roli detekce hran. Naopak při zvolení velké velikosti bloku se mohou umocňovat tmavá místa obrazu, proto je nutné volit jeho velikost v závislosti na velikosti QR kódu. [2]



Obrázek 4.3: Výstup prahování při nevhodně zvolené velikosti bloku prahování

Aby byla vystihnuta vhodná závislost velikosti bloku prahování na velikosti QR kódu, byl proveden v rámci návrhu experiment, díku kterému jsme zjistili, že velikost bloku prahování by měla odpovídat cca jedné pětině velikosti QR kódu. Experiment proběhl nad vzorkem obrazů o různých rozměrech, které byly tvořeny z větší části QR kódem. Tyto vzorky byly postupně prahovány s proměnlivými velikostmi prahovacího bloku a výsledky prahování porovnány s referenčními obrazy. Hodnota prahu byla vypočítávána aritmetickým průměrem hodnot okolí. Získanou závislost lze vidět na grafu 4.4.



Obrázek 4.4: Graf závislosti vhodné velikosti bloku prahování na rozměrech QR kódu

Jelikož velikost QR kódu do doby jeho detekce neznáme, bude nutné v rámci detekce QR kódu provést prahování s velikostí bloku odpovídající rozměrům vstupního obrazu či vhodným podílům rozměrů (pro vzdálenější QR kódy). Jakmile bude již QR kód detekován, bude známa jeho velikost, budeme moci aplikovat prahování podle experimentálně získané referenční závislosti.

4.4 Inverzní perspektivní transformace

Než se pustíme do popisu návrhu detekce a čtení QR kódu, je nutno definovat pojem a metodu inverzní perspektivní transformace, kterou budeme v následujících kapitolách používat.

Nutnost použití inverzní perspektivní transformace se odvíjí od způsobu interpretace QR kódu ve snímaném obraze, ve kterém je zobrazen v perspektivní projekci, v důsledku pozice kamery v prostoru. Abychom mohli QR kód zpracovat, musíme nalézt takový vztah mezi body QR kódu a jejich odpovídajícími body v perspektivní projekci, k čemuž slouží právě metoda homografie neboli inverzní perspektivní transformace.

Vztah mezi body QR kódu (x, y) a body promítaného QR kódu (X, Y) lze vyjádřit jako

$$p_{dst} = \mathbf{H} p_{src}, \tag{4.2}$$

$$\begin{pmatrix} xw \\ yw \\ w \end{pmatrix} = \begin{pmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}, \tag{4.3}$$

kde $\mathbf{H}$ je mapující maticí obsahující prvky $\vec{p} = (p_1, \ldots, p_9)^T$, tzv. stupně svobody, platí že $|\vec{p}| = 1$.

Pro zjištění závislosti mezi body v obou zobrazeních nám stačí určit vektor $\vec{p}$. Přičemž prvky vektoru $\vec{p}$ můžeme nalézt ze 4 nekolineárních bodů obrazu a jejich korespondujících

bodů v perspektivním zobrazení a řešením soustavy lineárních rovnic tak, aby platilo $|\vec{p}| = 1$ [20, 27]:

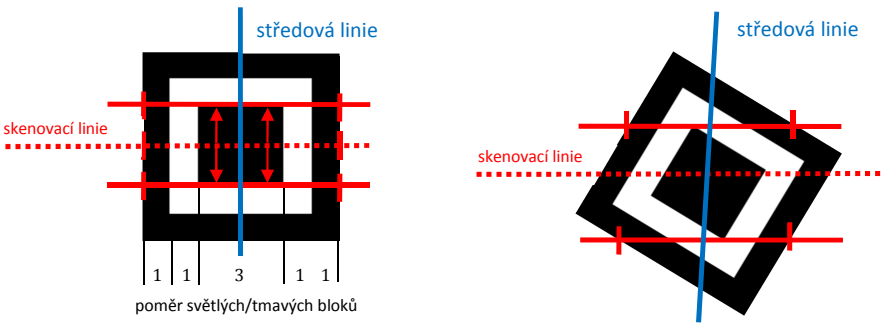
$$\mathbf{A}\vec{p} = \begin{pmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -X_1x_1 & -Y_1x_1 & x_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -X_1y_1 & -Y_1y_1 & y_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -X_2x_2 & -Y_2x_2 & x_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -X_2y_2 & -Y_2y_2 & y_2 \\ X_3 & Y_3 & 1 & 0 & 0 & 0 & -X_3x_3 & -Y_3x_3 & x_3 \\ 0 & 0 & 0 & X_3 & Y_3 & 1 & -X_3y_3 & -Y_3y_3 & y_3 \\ X_4 & Y_4 & 1 & 0 & 0 & 0 & -X_4x_4 & -Y_4x_4 & x_4 \\ 0 & 0 & 0 & X_4 & Y_4 & 1 & -X_4y_4 & -Y_4y_4 & y_4 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ p_8 \\ p_9 \end{pmatrix} = 0. \tag{4.4}$$

4.5 Detekce pozičních značek

Pro detekci pozičních značek norma zmiňuje použití skenovací linie, tato metoda je velmi efektivní a výhodná pro použití v reálném obraze. Ve svém principu využívá poměrů mezi tmavými a světlými moduly pozičních značek a je velmi dobře uplatnitelná u obrazů, ve kterých není QR kód v perspektivní projekci. K docílení čtení QR kódů i při velmi extrémním perspektivním zkreslení, kdy poměry mezi tmavými a světlými moduly pozičních značek jsou znehodnoceny, budeme detekci provádět pomocí hledání kontur (kapitola 4.5.2). Základní navržený princip vycházející z normy pomocí skenovací linie si přesto popíšeme v kapitole 4.5.1.

4.5.1 Skenovací linie

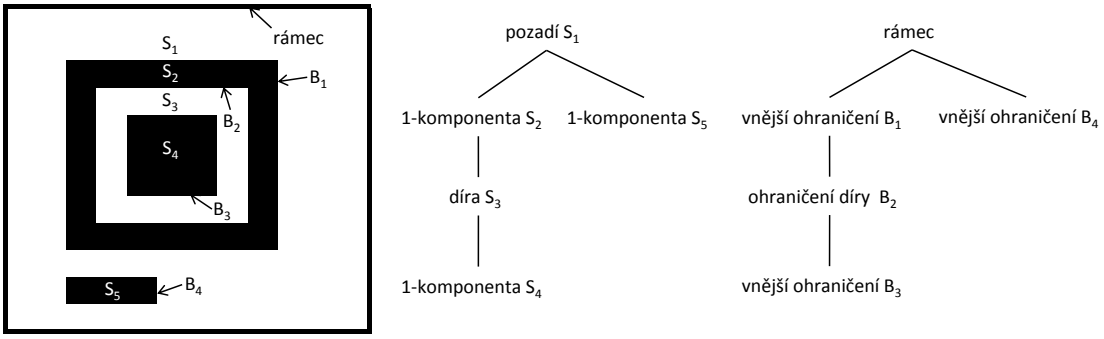
Princip hledání pozičních značek s využitím skenovací linie lze definovat procházením jednotlivých řádků obrazu a hledáním takové sekvence tmavých a světlých bodů, která by odpovídala poměru 1:1:3:1:1. Akceptovaná nepřesnost v poměru je dána podle normy hodnotou 0.5, takže lze např. přijmout i sekvenci v poměru 0.5:1.5:2.5:0.5:1.5. Jakmile tuto sekvenci bodů skenovací linie nalezne (obrázek 4.5), dojde k jejímu rozšíření ve vertikálním směru nahoru i dolů, dokud není dosaženo hranice vnitřního tmavého čtverce poziční značky. Střed poziční značky poté můžeme zjistit z průsečíků středových linií. První středová linie je zkonstruována ze středů úseček, ležících na rozšířených skenovacích liniích, definovaných hranicemi poziční značky. Ke zjištění druhé středové linie musíme využít skenovací linie ve vertikálním směru a obdobného postupu.



Obrázek 4.5: Detekce poziční značky pomocí skenovací linie

4.5.2 Hledání kontur

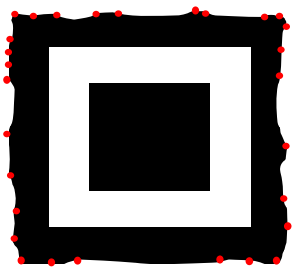
Hledání kontur slouží ke strukturální analýze obrazu. Algoritmy na hledání kontur pracují většinou s již binarizovanými obrazy a jejich princip spočívá ve hledání spojitých komponent, které jsou tvořeny buď 1 pixely (tzv. 1-komponenty) nebo 0 pixely (tzv. 0-komponenty či díry). Nejpoužívanějšími algoritmy jsou Square tracing, Moore-Neighbor, Radial Sweep aj. Vyjma těchto algoritmů existují i algoritmy, které dokáží rozpoznat a uchovat topologii mezi nalezenými strukturami, jedním z nich je např. Suzuki 85 algoritmus [28].



Obrázek 4.6: Výstup nálezu kontur a sestavení topologické závislosti mezi nimi

Využití topologie by mohlo vést ke zrychlení detekce pozičních značek, neboť víme, že poziční značka musí být tvořena právě jednou další 1-komponentou (obrázek 4.6), čehož lze využít k rychlému zahazování neplatných struktur. Topologie by šla využít i pro detekci a zpracování více QR kódů v obraze, kde by ovšem musel být každý QR kód ohraničen tzn. být ve vlastní struktuře. Identifikace a přiřazení pozičních značek k jednotlivým QR kódům by poté probíhalo na základě shody rodičovské struktury. Právě z těchto důvodů bude v implementaci využito topologické analýzy a Suzuki algoritmu.

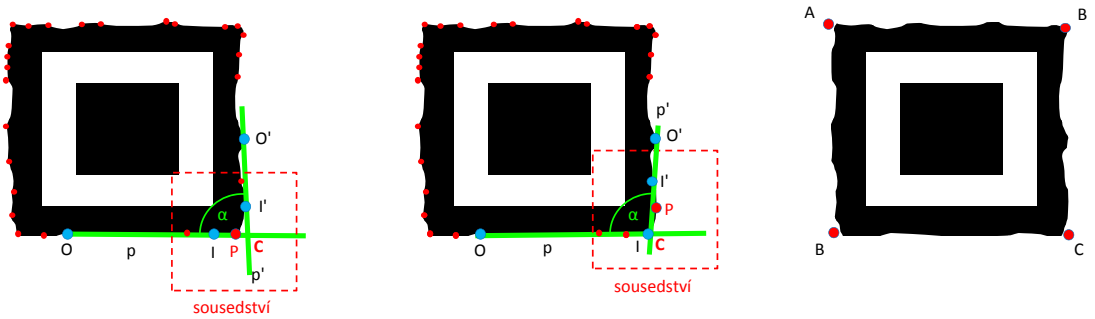
Výsledkem hledání kontury bývá vždy množina bodů, která sleduje danou strukturu, v našem případě např. poziční značku. Pokud uvážíme již i nějaký druh interpolace bodů, možný výsledek detekce poziční značky by mohl vypadat tak, jak je znázorněno na obrázku 4.7.



Obrázek 4.7: Předpokládaný interpolovaný výstup bodů hledání kontury pro poziční značku

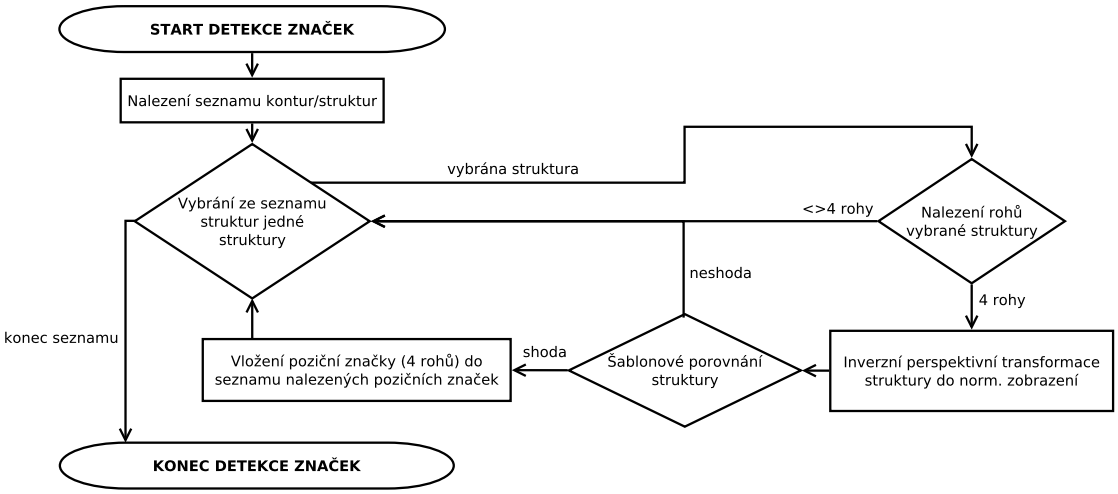
Poziční značka, jako obrazec čtvercovitého charakteru, je přesně definována čtyřmi body (rohy). Množina bodů kontury nám ovšem tuto informaci přímo nedává, ba dokonce rohy v kontuře mohou být různě zdeformované, proto je nutno skutečné, resp. pomyslné rohy

poziční značky zjistit jiným způsobem. Pro tento účel byl navržen algoritmus hledání rohů polygonu, jehož přesné znění lze nalézt v příloze C a vizualizaci jeho činnosti na následujícím obrázku 4.8.



Obrázek 4.8: Výstup algoritmu hledání rohů polygonu (zde poziční značky)

Pokud algoritmus hledání rohů polygonu skončí se čtyřmi nalezenými rohy, můžeme se domnívat, že se jedná o poziční značku a provést další potřebné kroky k jejímu ověření, v opačném případě lze danou strukturu zahodit a přejít na zpracování další struktury. Při nálezu čtyř rohů přistoupíme k inverzní perspektivní transformaci dané struktury na základě těchto bodů do normalizovaného zobrazení. Šablonovým porovnáním poté zjistíme, zda se jedná o poziční značku či ne a přejdeme na zpracování další struktury. Výsledný vývojový diagram celého procesu detekce lze vidět na následujícím obrázku 4.9.

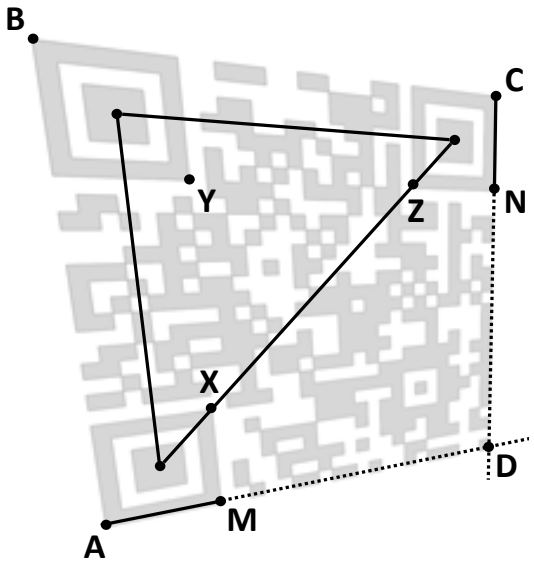


Obrázek 4.9: Proces detekce pozičních značek QR kódu

4.6 Zjištění a transformace oblasti QR kódu

Proces detekce pozičních značek nám bude vracet seznam detekovaných značek, kde každá značka bude přesně definována čtyřmi body seřazenými ve směru hodinových ručiček. Abychom mohli takto detekovaných značek využít pro převod QR kódu do normalizovaného zobrazení, budeme muset nejprve poziční značky a jejich body uspořádat a identifikovat jejich umístění v QR kódu. Jelikož se poziční značky nachází v QR kódu pouze tři, ve třech rozích, a pro inverzní perspektivní transformaci je zapotřebí čtyř bodů, budeme muset určit i tento čtvrtý bod (roh QR kódu).

Postup seřazení pozičních značek (obrázek 4.10) by mohl být založen na vytvoření pomyslného trojúhelníku ze středů pozičních značek, výpočtu jeho těžiště, díky němuž by šlo poziční značky uspořádat ve směru hodinových ručiček. Tímto seřazením by nám ale stále scházela informace o pozici značek a jejich bodů v rámci QR kódu, tuto informaci by šlo získat na základě nalezení bodů pozičních značek X, Y, Z, které se nachází uvnitř QR kódu. Vnitřní bod každé poziční značky by byl nalezen otestováním všech bodů dané poziční značky a zjištěním takového bodu, který je nejvzdálenějším bodem ke vnitřní stěně našeho pomyslného trojúhelníku nebo nejbližším bodem ke vnější straně trojúhelníku, neexistuje-li žádný bod poziční značky uvnitř trojúhelníku. Poté můžeme tvrdit, že levá horní poziční značka je taková značka, jež obsahuje nejvzdálenější vnitřní bod Y poziční značky od vnitřní strany pomyslného trojúhelníku.

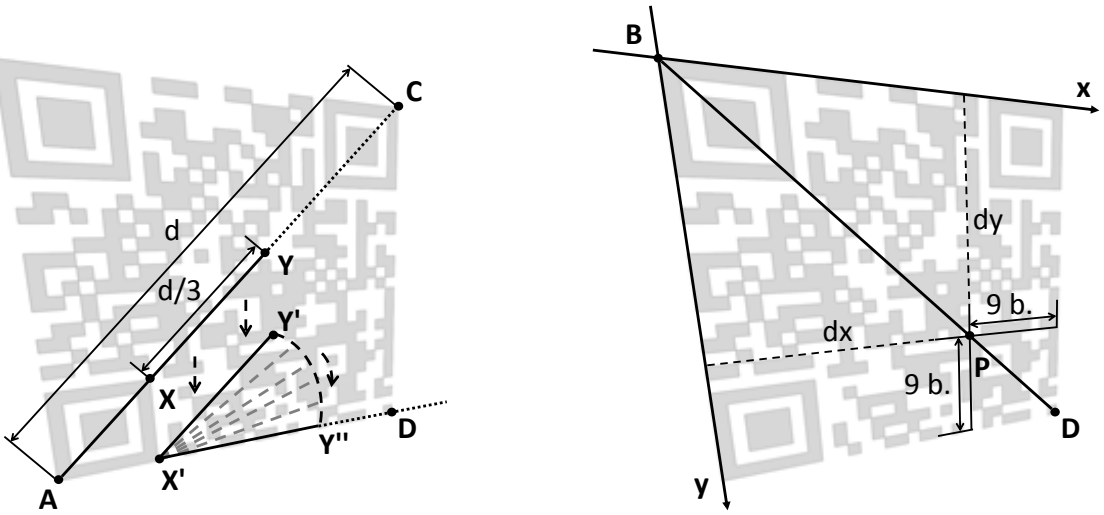


Obrázek 4.10: Znázornění postupu zjištění čtyř bodů pro inverzní perspektivní transformaci

Po seřazení bodů pozičních značek již budeme moci určit i čtyři body pro inverzní perspektivní transformaci A, B, C a D, kde poslední bod D potřebný pro transformaci by bylo možné v případě ideálně sejmutého QR kódu získat protnutím polopřímek AM a CN. V praxi se ovšem můžeme setkat i s nepřesně detekovanými pozičními značkami, proto byly navrženy další dva postupy získání 4. bodu (rohu) QR kódu (obrázek 4.11).

První navržený postup hledání 4. bodu je založen na konstrukci vzorkovací úsečky (obdélníku) $X'Y'$, která vzorkuje QR kód postupnou rotací z počátečního bodu X' ve směru hodinových ručiček, dokud není dosaženo světlého pozadí, které kód obklopuje. Po skon-

čení vzorkování můžeme rozšířením vzorkovací úsečky $X'Y''$ získat polopřímku, na které bude ležet bod D. Pokud totéž provedeme i vůči pravé horní poziční značce a proti směru hodinových ručiček, tak bod D získáme právě z průsečíku námi rozšířených vzorkovacích úseček.



Obrázek 4.11: Zjištění 4. bodu vzorkováním (vlevo) a pomocí zarovnávací značky (vpravo)

Druhý postup hledání 4. bodu není použitelný pro QR kódy verze 1, které neobsahují zarovnávací značky. Principiálně ke své činnosti využívá jiné z předešlých metod pro převod do dočasného normalizovaného zobrazení, ve kterém nalezne pozici bodu D pomocí pravé dolní zarovnávací značky a poté opět zpětně tento bod transformuje do perspektivního zobrazení.

K nalezení skutečné pozice P zarovnávací značky v normalizovaném zobrazení využijeme šablonového vyhledávání, abychom ale mohli vytvořit šablonu, musíme nejprve vhodně zjistit velikost jednoho modulu. Pro výpočet pozice bodu D bude nutno také zjistit referenční pozici s zarovnávací značky (viz ISO norma tabulka E.1), což zahrnuje i přečtení verze QR kódu. Při znalosti, že bod D je od zarovnávací značky vždy vzdálen 9 bodů oběma směry, platí:

$$D = \left[B_x + \frac{dx}{s}(s+9); B_y + \frac{dy}{s}(s+9) \right], \tag{4.5}$$

$$dx = P_x - B_x, \quad dy = P_y - B_y. \tag{4.6}$$

Po získání bodu D v normalizovaném zobrazení stačí již pouze tento bod převést zpátky do perspektivního zobrazení. Tento postup hledání 4. bodu byl navržen z důvodu umožnění čtení QR kódů na znečištěném podkladu, což by vedlo k selhání první uvedené metody.

4.7 Čtení informací z QR kódu

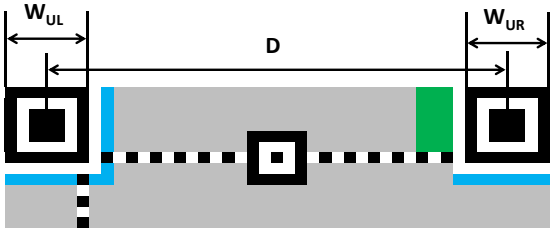
Pro čtení QR kódu bude implementována čtecí mřížka (obrázek 4.12), která bude na celý QR kód přikládána. Jednotlivé moduly QR kódu reprezentující bity budou vzorkovány ze středu každé buňky přiložené mřížky. Vzorkovaná oblast bude zabírat 3x3 pixely, jak je uvedeno v referenčním algoritmu ISO normy.

Obrázek 4.12: Ustanovení vzorkovací mřížky a oblastí vzorkování (červená barva)

O počtu řádků a sloupců mřížky bude rozhodnuto v závislosti na detekované verzi V QR kódu (obrázek 4.13), která bude zprvu určena vtahem (dle ISO normy)

$$V = \frac{7D}{2(W_{UL} + W_{UR})} - 2,5. \tag{4.7}$$

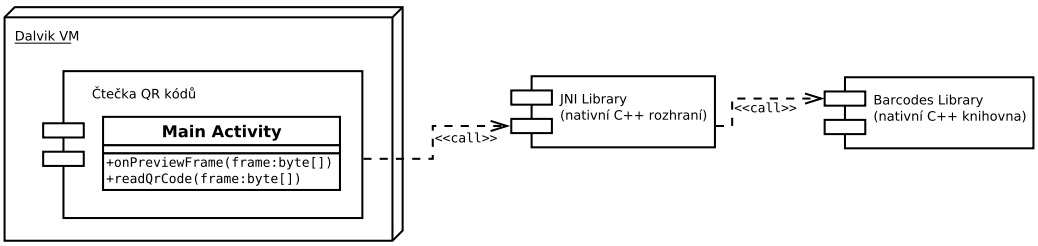
Bude-li detekován QR kód verze 6 nebo nižší, bude tato verze považována jako finální. V opačném případě dojde k upřesnění verze V přečtením metainformace o verzi z QR kódu.



Obrázek 4.13: Zjištění dočasné verze QR kódu podle jeho rozměrů

4.8 Architektura cílové aplikace

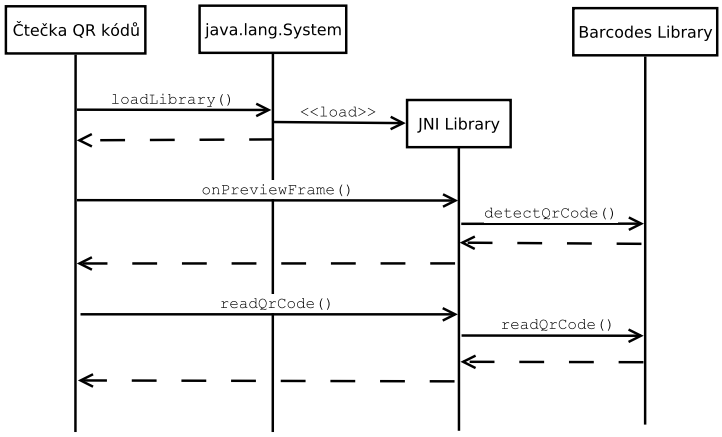
Aplikace čtečky bude implementována pro platformu Android s ohledem na případné znovuvyužití stěžejních algoritmů zajišťujících čtení QR kódu i na ostatních mobilních platformách. Nejenom z tohoto důvodu, ale také i z efektivního využití výpočetních prostředků, bude implementována detekce a čtení QR kódu v nativním kódu jazyka C++. Sestavené C++ moduly budou poté ve formě sdílené knihovny (`Barcodes Library`) linkovány do naší aplikace, ve které bude nutno prostřednictvím JNI[1] implementovat rozhraní pro komunikaci s naší knihovnou z jazyka Java (`JNI Library`) (obrázek 4.14).



Obrázek 4.14: Knihovní závislosti aplikace čtečky QR kódů

Z důvodu snadné rozšiřitelnosti aplikace o nově čitelné QR kódy, bez potřeby překládání nativní knihovny, bude knihovna `Barcodes Library` obsahovat pouze dekodér fyzické vrstvy, který je pro všechny případy použití QR kódu neměnný. Dekodér aplikační vrstvy bude implementován až v rámci aplikace v jazyce Java. Jelikož nám Dalvik dovoluje dynamické načítání tříd pomocí `DexClassLoader`, bude zvážena i možnost instalování dekodérů aplikační vrstvy za běhu aplikace ve formě zásuvných modulů.

Rozhraní mezi nativní knihovnou a aplikací bude obsahovat řadu obalujících objektů, které budou zprostředkovávat převod C++ objektů na Java objekty a naopak. Aplikací budou volány z rozhraní pouze dvě funkce, jedna pro detekci QR kódu v reálném obraze a druhá pro dekódování QR kódu ze sejmutého obrazu (obrázek 4.15).



Obrázek 4.15: Sekvenční diagram znázorňující volání funkcí nativního rozhraní

[1] Java Native Interface – Rozhraní umožňující oboustranné propojení Java kódu, interpretovaným na virtuálním stroji, s kódem nativním, jako je C++, C, assembler apod.

Kapitola 5

Implementace

Následující kapitola pojednává o samotné realizaci algoritmů a navržené aplikaci čtečky QR kódů (kapitola 4). Popis implementace je zaměřen na uvedení nejzákladnějších tříd, metod či funkcí, které implementují nejdůležitější části výsledné aplikace, přičemž podrobný popis lze nalézt v programových dokumentacích na přiloženém DVD (příloha A).

5.1 Knihovna pro čtení QR kódů

Knihovna pro rozpoznání QR kódů (kapitola 5.1.1), čtení a dekódování jejich fyzické vrstvy (kapitola 5.1.3) byla implementována v jazyce C++. Při práci s obrazem bylo využito open source knihovny OpenCV. Strukturálně byla knihovna realizována tak, aby ji bylo možné v budoucnu případně rozšířit o nově podporované čárové kódy.

5.1.1 Detekce QR kódu

K detekci QR kódu v obraze slouží singleton třída `QrDetector` a její metoda `detect()`, jejíž parametry jsou:

- `image` – vstupní zpracovávaný obraz, který je již reprezentován ve stupních šedi;
- `detectedMarks` – výstup vektor s detekovanými rohy pozičních značek QR kódu v obraze;
- `flags` – doplňující informace pro detekci ve formě bitových vlajek.

Výsledek činnosti metody detekce velmi předurčuje právě poslední parametr `flags`, jenž slouží zejména pro specifikaci předpokládané vzdálenosti QR kódu v obraze. Zadaná vzdálenost QR kódu ovlivňuje výpočet velikosti bloku prahování, tzn. výsledný binarizovaný obraz, kde budou detekovány poziční značky.

Byly zvoleny celkem čtyři předpokládané vzdálenosti QR kódu, které mohou být předány metodě i současně. Jedna ze vzdáleností odpovídá velmi blízkému QR kódu a pokud je předána, tak se pro výpočet velikosti bloku prahování uplatní experimentálně zjištěná referenční závislost (kapitola 4.3 obrázek 4.4). Ostatní tři vzdálenosti jsou určeny pro vzdálenější QR kódy, konkrétně pro QR kódy o velikosti 1/3, 1/6 nebo 1/12 celkového obrazu. Velikost bloku prahování v tomto případě je dána stejnou poměrnou částí referenční závislosti.

Pokud předáme metodě více předpokládaných vzdáleností QR kódu, budou algoritmem detekce preferovány nejprve ty menší, nebot' lze předpokládat, že QR kód budeme číst větši-

nou z blízké vzdálenosti. Nenalezne-li algoritmus alespoň tři poziční značky v nejmenší předané vzdálenosti, zkusí vybrat další větší předpokládanou vzdálenost QR kódu. Tento proces algoritmus opakuje, dokud nejsou vybrány všechny předpokládané vzdálenosti z `flags` nebo nejsou nalezeny alespoň tři poziční značky. Jelikož během detekce, která běží v několika krocích, může dojít k detekci stejných značek, bylo nutné implementovat i mechanismus filtrování těchto značek.

Proces detekce byl implementován podle navrženého postupu:

1. **Nalezení všech struktur** v obraze pomocí hledání kontur Suzuki 85 algoritmem a funkcí `findContours()`, kterou již implementuje knihovna OpenCV.
2. **Zjištění rohů struktur** metodou `Polygon2D::findCorners()` (příloha C) a zahození struktur, pokud u nich nebyly nalezeny přesně 4 rohy.
3. **Převedení obrazu** do normalizovaného zobrazení funkcí `warpPerspective()`.
4. **Šablonové porovnání** struktury se šablonou poziční značky funkcí `exactMatch()`.

5.1.2 Optimalizace detekce pozičních značek

Z důvodu urychlení detekce pozičních značek, tzn. vyhnutí se časově drahému převodu obrazu z perspektivního zobrazení do normalizovaného, byly přidány s využitím OpenCV do již implementovaného algoritmu detekce další tři kroky. Svojí funkcí plní jakési filtry, které mají za úkol zahazovat struktury, jež nejsou s velkou pravděpodobností pozičními značkami.

První filtr je založen na nalezení nejmenšího ohraničujícího obdélníku (obrázek 5.1) dané struktury. Filtr slouží pro zahazování segmentů, jejichž ohraničující obdélník má příliš malé rozměry nebo je velký poměrový rozdíl velikostí šířky a délky tohoto obdélníku. V ideálním případě poziční značky je poměr mezi šířkou a délkou 1:1. Jelikož poziční značka může být perspektivně zkreslena, byl zvolen jako maximální přípustný poměr 1:4.

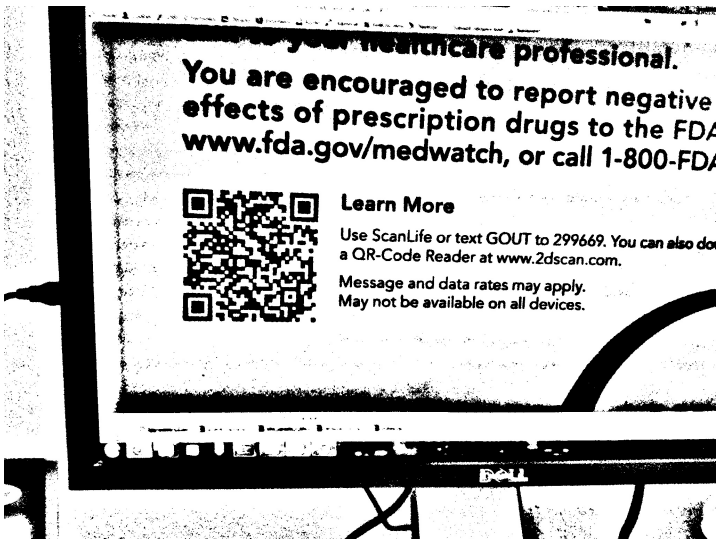
Obrázek 5.1: Nejmenší ohraničující obdélník (vlevo) a konvexní obálka (vpravo) struktury

Druhý filtr, který byl zařazen do algoritmu, provádí analýzu poměru tmavých a bílých pixelů struktury. V ideálním případě struktury poziční značky je tento poměr 49:16. V rámci detekce ale tolerujeme až neshodu v 8 bodech, tudíž akceptujeme i poměry 57:8 či 41:24.

Třetí filtr vytvoří kolem detekované struktury konvexní obálku (obrázek 5.1) a zjistí počet pixelů uvnitř této obálky. S využitím předchozího filtru, kde jsme získali počet pixelů struktury, můžeme obě hodnoty porovnat. Jelikož tvar poziční značky je přesně určen pouze svými 4 nekolineárními body a součet vnitřních úhlů značky je 360°, nikdy nemůže u struktury ideální poziční značky nastat, že by některý z vnitřních úhlů byl konkávní. Konkávnost

vnitřních úhlů je možné získat až pro 5 a více bodů. I když počet bodů konvexní obálky a struktury poziční značky by měl být totožný, implementačně je tolerována až 10% neshoda.

Pokud budeme prezentovat funkci filtrů na ukázkovém obrazu (obrázek 5.2), jenž obsahuje celkem 7000 nalezených struktur, filtry připustí k otestování na poziční značku pouze 21 z nich. Většina struktur je odstraněna již v prvním filtru, konkrétně 6742. Dalších 79 struktur neprojde druhým filtrem a zbývajících 158 struktur třetím filtrem.



Obrázek 5.2: Ukázka obrazu se 7000 strukturami

Pro optimalizaci detekce pozičních značek v reálném obraze byly přidány další dva topologické filtry. Ty jsou ovšem volitelné a provádí se pouze při předání specifické bitové vlajky parametrem `flags` metodě detekce. Jejich použití při detekci užité během dekódování by bylo nevhodné, pokud by byl QR kód např. poškozen a topologická závislost ztracena. První filtr vychází z faktu zmíněného již během návrhu, že poziční značka musí být tvořena alespoň jednou další strukturou (menší soustředný čtverec). Druhý filtr je aplikován po detekci první poziční značky a zjištění její rodičovské struktury, která musí být společná pro všechny poziční značky.

5.1.3 Čtení QR kódu

Ke čtení QR kódu v obraze slouží singleton třída `QrDecoder` a její metoda `decode()`, jejíž parametry jsou:

- `image` – vstupní zpracovávaný obraz, který je již ve stupních šedi;
- `dataSegments` – výstupní vektor s přečtenými datovými segmenty;
- `flags` – doplňující informace ve formě bitových vlajek.

Pro čtení byla vytvořena řada pomocných tříd, do kterých byla rozložena vhodně funkčnost. V jednotlivých krocích čtení se uplatňuje metodika dekódování uvedená v ISO normě. Jmenujme si proto tyto kroky, v pořadí v jakém jsou aplikovány, a odpovídající nejzákladnější třídy a metody, které funkčnost implementují.

Implementace kroků dekódování

1. **Získání oblasti QR kódu** pro jeho inverzní perspektivní transformaci je prováděno pomocí funktorů, jejichž názvy tříd začínají na „`PerspCornersFrom`“. Byly implementovány celkem tři funktory, podle navržených postupů. Selže-li dekódování s jedním funktorem, je vybrán další.

2. **Transformace QR kódu** do normalizovaného zobrazení je zprostředkována funkcí `warpPerspective`. Po jejím dokončení je nutno QR kód opět binarizovat statickou metodou `QrDetector::binarize()`. Binarizace tentokrát proběhne podle experimentálně získané referenční závislosti a nikoliv v závislosti na předpokládané vzdálenosti QR kódu v obraze. V normalizovaném zobrazení QR kód tedy vyplňuje celý obraz.

3. **Zjištění verze QR kódu** z obrazu je důležité pro sestrojení čtecí mřížky, je implementováno třídou `QrVersionInformation` a statickou metodou `fromImage()`, která vrací instanci této třídy.

4. **Získání modulů QR kódu** při znalosti verze QR kódu se provádí „přiložením“ čtecí mřížky nad QR kód ve statické metodě `BitMatrix::fromImage()`, díky níž získáme reprezentaci QR kódu v matici booleovských hodnot.

5. **Zjištění formátu QR kódu** je docíleno již z bitové matice získané v předešlém kroku a implementováno statickou metodou `QrFormatInformation::fromBitMatrix()`, která vrací podobně jako u verze instanci této třídy.

6. **Maskování dat QR kódu** je uskutečněno sestavením masky podle velikosti a formátu QR kódu instancí třídy `QrFormatInformation` a metodou `buildXORDataMask()`. Operaci maskování implementuje třída `BitMatrix`.

7. **Extrakce sekvence dat** a korekce chyby z QR kódu je zařízena metodou `sample()` třídy `GridSampler`, která vzorkuje bitovou matici menší maticí, v našem případě o velikosti kódového slova QR kódu. Sekvence je získána z kódu ve formě bitového pole `BitArray`.

8. **Uspořádání sekvence dat** a korekce chyby do bloků je zapotřebí z důvodu prokládání kódových slov a zpřístupnění opravy chyb, která probíhá vždy nad těmito menšími bloky. Za tímto účelem byla vytvořena třída `QrCodewordOrganizer`.

9. **Oprava případných chyb** v jednotlivých blocích je realizována s pomocí open source Reed-Solomon knihovny napsané v jazyce Java [12], která byla do jazyka C++ portována. K opravení chyb můžeme přistoupit prostřednictvím třídy `QrReedSolomon`.

10. **Dekódování dat** z fyzické vrstvy QR kódu je umožněno třídou `QrBitDecoder`, která shromažďuje a v případě potřeby využívá instance implementovaných dekodérů různých režimů. Při dekódování se využívá proudu bitů zprostředkovaného třídou `BitStream`, přičemž bitový proud je ustanoven nad přečteným polem bitů `BitArray`.

5.2 Aplikace čtečky QR kódů

Aplikace čtečky QR kódů byla implementována v jazyce Java pro cílovou platformu Android. Ke své činnosti využívá výše popsanou C++ knihovnu (kapitola 5.1), se kterou komunikuje skrze JNI.

Uživatelsky viditelná část aplikace se skládá ze sedmi Activity komponent (kapitola 5.2.1). Z důvodu popsaného v návrhu implementuje aplikace i mechanismus dekódování aplikační vrstvy QR kódů (kapitola 5.2.2), který je postaven na možnosti snadného přidávání nových dekodérů do aplikace i včetně skrze zásuvné moduly.

Jelikož některé akce v aplikaci využívají veřejných Intent požadavků aplikace OI File Manager, je doporučené mít tuto aplikaci pro zkvalitnění práce se čtečkou QR kódu nainstalovanou.

5.2.1 Komponenty aplikace

Aplikace je tvořena převážně Activity komponentami. Přehled, s jakými třídami komponenty kooperují, můžeme získat z diagramu tříd v příloze D.

Hlavní spouštěcí Activity komponentou je `MainActivity`, která v sobě implementuje funkčnost zobrazení náhledu z kamery a volání nativních funkcí skrze JNI rozhraní. Probíhá zde jak detekce, tak i dekódování QR kódu z obrazu, přičemž obě tyto činnosti běží ve svém vlastním vlákně. V případě detekce je z kamery přebírán obraz ve formátu NV21, který je převáděn v nativní funkci do stupňů šedi. Jelikož je NV21 nejčastějším formátem pro náhled, se kterým se u zařízení pracujících na platformě Android můžeme setkat, nejsou žádné další formáty aktuálně podporovány. Obraz pro dekódování QR kódu je získáván z callback funkce volané z API Androidu při požadavku o sejmutí obrazu. Tato funkce vrací komprimovaný obraz ve formátu JPEG, který je opět v nativní funkci rozhraní převáděn do stupňů šedi.

Pokud `MainActivity` komponenta nalezne QR kód a dekóduje jeho fyzickou vrstvu, jsou obraz a serializovaný objekt obsahující data QR kódu uloženy do souborů v privátní složce aplikace a je zavolána komponenta `OpenQrActivity`, které jsou Intent požadavkem předána umístění obou souborů. Komponenta `OpenQrActivity` slouží pro dekódování aplikační vrstvy a zobrazení obrazovky s výsledkem dekódování. Problematika dekódování je blíže popsána v následující kapitole 5.2.2.

Dalšími spíše doplňkovými Activity komponentami jsou:

- `SettingsActivity` – implementuje nastavení aplikace, zejména škálovatelnost rozlišení a kvality obrazu pro náhled z kamery a snímaného obrazu v JPEG formátu. Dalšími užitečnými volbami může být zapnutí automatického zaostřování nebo blesku pro snímání obrazu aj.;
- `InstallationManagerActivity` – slouží pro správu a instalování nových zásuvných modulů;
- `BrowseImagesActivity` a `BrowseQRCodesActivity` – komponenty jsou určeny pro procházení uložených QR kódů a sejmutých obrazů, příp. pro jejich správu skrze OI File Manager;
- `AboutActivity` – komponenta zobrazuje informativní obrazovku o aplikaci.

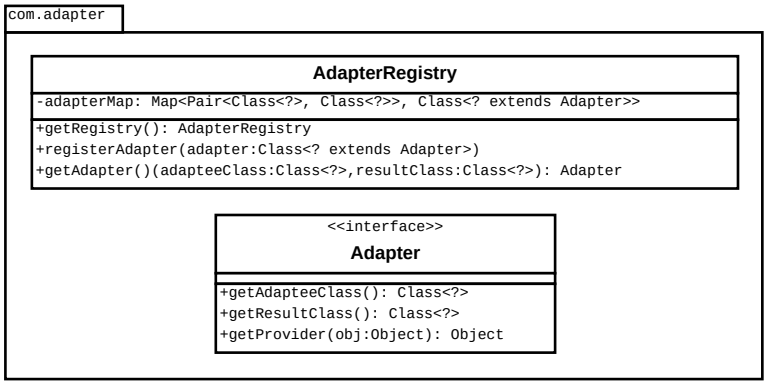
5.2.2 Dekódování aplikační vrstvy QR kódů

Dekódování aplikační vrstvy QR kódu se provádí po předání kódu Intent požadavkem komponentě `OpenQrActivity`. V ní je dále volána ze singleton třídy `QrDecoderManager` metoda `decode()`, která vrací objekt reprezentující specifický QR kód.

Proces dekódování je založen na postupném vybírání všech dostupných dekodérů a testování dekódování pomocí každého z nich, dokud není QR kód úspěšně dekódován. K urychlení a zamezení nevhodnému testování dekodérů byla přidána možnost informování o podporovaných URI schématech, kterými dekodér disponuje. Pokud je ovšem dekodér „binární“ a nepodporuje URI schémata, nic jiného než testování nám nezbývá.

Aktuálně jsou v aplikaci zahrnuty pouze dva dekodéry `BaseQrDecoder` a `TextDecoder`. `BaseQrDecoder` implementuje dekódování následujících URI schémat: `http`, `mailto`, `tel`, `URLTO`, `sms`, `SMSTO`. `TextDecoder` není dekodérem URI schémat a slouží pro dekódování prostého textu, tudíž má i nejnižší prioritu ze všech dekodérů, nebot' většina dnešních přenosových protokolů je právě textového formátu.

Po dokončení dekódování bylo nutné řešit reprezentaci získaného objektu QR kódu na obrazovce v podobě pohledu `View`. Pro tento účel byl implementován registr adaptérů `AdapterRegistry` (obrázek 5.3). Adaptéry jsou zde určeny k převodu specifického QR kódu na jeho odpovídající pohled, přičemž chceme-li daný adaptér vyhledat, musí být registrován v registru. Nejpodstatnější metodou adaptéru je metoda `getProvider()`, která vrací samotný objekt zprostředkovávající převod, konkrétně objekt implementující rozhraní `QrCodeViewProvider`. Celkově byly implementovány adaptéry pro zobrazení QR kódů obsahujících SMS, e-mail, tel. číslo, internetové stránky a prostý text.



Obrázek 5.3: Registr adaptérů využitý pro mapování QR kódů na odpovídající pohledy

Pro snadné přidávání dekodérů aplikační vrstvy, pouhým rozšířením pole dostupných dekodérů, a pohledů QR kódů, zaregistrováním v registru adaptérů, byla přidána podpora zásuvných doplňků v aplikaci.

Zásuvný doplněk je prostý JAR archiv, který obsahuje zkompilované třídy ve formátu Dalvik Executable a ve složce `res` dva popisné XML soubory. První XML soubor `package.xml` sděluje informace o doplňku, aktuálně pouze jeho název a základní popis. Druhý XML soubor `classes.xml` říká třídě spravující doplňky `InstallationManager`, které třídy jsou obsažené v archivu a mají být načteny. Správce doplňků dále využívají specializované třídy pro načítání nových dekodérů `InstallableQrDecoderManager` a pro načítání pohledů `InstallableQrCodeViewManager`.

Kapitola 6

Dosažené výsledky

V této kapitole bude zhodnocena výkonnost a spolehlivost implementovaného řešení detekce a čtení QR kódů. Závěrem budou diskutována budoucí možná rozšíření aplikace.

Pro vyhodnocení spolehlivosti a výkonnosti aplikace byly použity výbavou velmi odlišné mobilní telefony Samsung S5570 Galaxy Mini (dále pouze „Samsung“) a LG P990 Optimus 2X (dále pouze „LG“). Telefon LG, díky svému dvoujádrovému procesoru a kvalitnímu 8 Mpix fotoaparátu, dosáhl v následujících testech značně lepších výsledků než telefon Samsung.

6.1 Vyhodnocení spolehlivosti

Spolehlivost čtečky QR kódů je dána především kvalitou vstupního obrazu. Bylo zjištěno, že největší negativní vliv na čtení má rozostřený obraz. Bílé moduly QR kódu jsou potom během binarizace umocňovány, což způsobuje někdy až ztrátu modulů tmavých. Ačkoliv ztracené moduly v QR kódu mohou být často opraveny, problém nastává při neostrých pozičních značkách, které musí splňovat pro úspěšnou detekci 80% shodu v šablonovém porovnání. Při čtení QR kódu z malé vzdálenosti se ale s předchozími problémy nesetkáme. Výjimku tvoří pouze QR kódy nejvyšších verzí, kde poziční značky zabírají velmi malou část obrazu i z blízka sejmutého obrazu. Takové kódy se ovšem v praxi téměř vůbec nepoužívají.

Čtečka si umí dobře poradit s QR kódy, které jsou sejmuty při špatných světelných podmínkách. Pokud je QR kód poškozen, je použita Reed-Solomonova metoda k jeho opravení (obrázek 6.1).

Obrázek 6.1: Čitelný a opravitelný QR kód

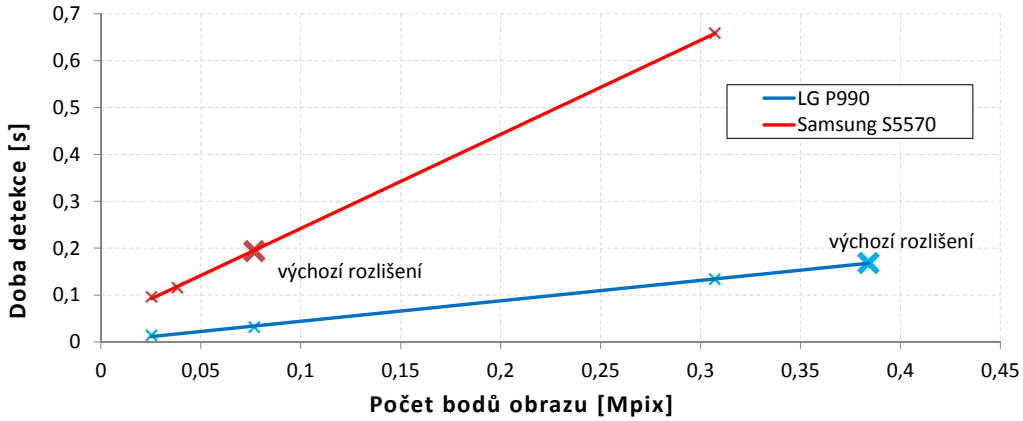
Při detekci QR kódu a transformaci do normalizovaného zobrazení není čtečka závislá na nalezení zarovnávacích značek, tak jako všechny ostatní testované čtečky QR kódů [5, 13, 11]. Selže-li ovšem čtení s jednou metodou transformace, je využito i metody založené na nálezu zarovnávacích značek. Nachází-li se v obraze více QR kódů, čili obsahuje více než 3 poziční značky, snaží se čtečka tyto poziční značky uskupit tak, aby alespoň jeden z kódu mohl být dekódován. Podmínkou ovšem zůstává, že kódy musí být ohraničeny. Čtečkou mohou být čteny i vzdálené QR kódy, musí být ale splněna minimální velikost stran minimálního ohraničujícího obdélníku poziční značky (kapitola 5.1.2). Pro čtení tato velikost je 14 pixelů.

Naopak čtečka si neumí poradit s QR kódy, které jsou např. vylepené na zakřiveném povrchu láhve. S telefonem Samsung se nepodařilo řádně přečíst QR kód nejvyšší verze, pouze ho detekovat. Telefon LG si s tímto kódem poradil, ale pouze až s využitím automatického zaostření. Jakékoliv porušení struktury poziční značky, vedoucí např. k nalezení 5. rohu poziční značky nebo spojení dvou struktur, způsobí neúspěšnou detekci QR kódu v obraze.

6.2 Vyhodnocení výkonnosti

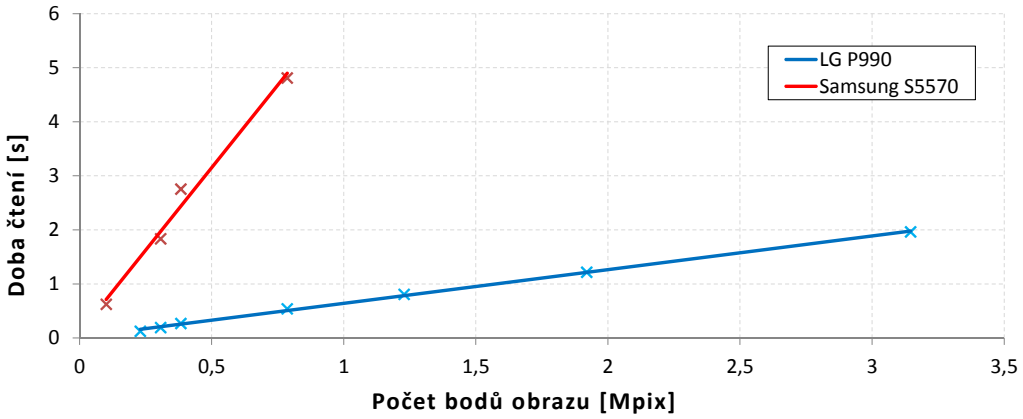
Při vyhodnocování výkonnosti byla zkoumána celková doba, jaká je potřebná pro provedení detekce QR kódu v reálném obraze a jeho čtení v sejmutém obraze. Hlavním faktorem ovlivňujícím výkonnost bylo dle očekávání rozlišení zpracovávaného obrazu. Referenční použitý obraz pro testování obsahoval neporušený QR kód verze 3.

Ke změření závislosti doby detekce pozičních značek na velikosti obrazu bylo proměnlivě nastavováno rozlišení náhledu z kamery, který je k detekci využíván. Doba byla odečítána pomocí systémového času, nikoliv procesorového, tudíž ji lze použít i k příp. převodu na zpracované rámce za sekundu. Získanou závislost lze vidět v grafu na obrázku 6.2. Při výchozím nastavení rozlišení náhledu, tzn. rozlišení odpovídajícímu rozlišení displeje, bylo u obou přístrojů dosaženo frekvence obnovy volání detekce 5-6 rámců za sekundu. Jediným faktorem, který může frekvenci v daném rozlišení výrazně ovlivnit, je počet detekovaných struktur v rámci metody detekce. Není-li obraz strukturovaný, lze pozorovat při výchozím rozlišení zpracování až 10 rámců obrazu za sekundu, v opačném případě naopak výrazně méně.



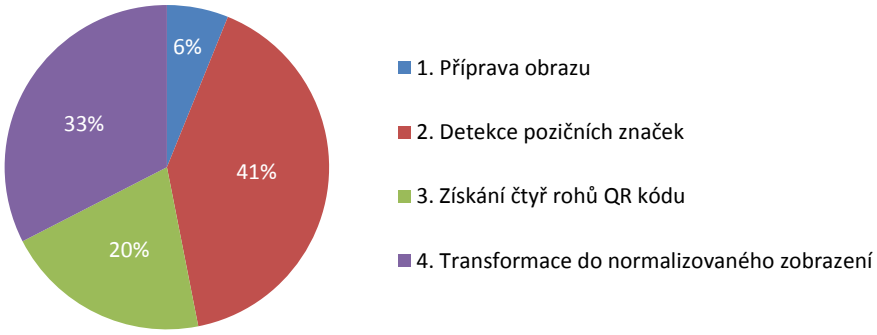
Obrázek 6.2: Graf závislosti doby detekce pozičních značek na velikosti obrazu

Vyhodnocení výkonnosti čtení probíhalo obdobným způsobem jako u detekce pozičních značek, pouze s tím rozdílem, že rozlišení nebylo proměnlivě nastavováno náhledu z kamery, ale z kamery sejmutému obrazu. Výkonnost čtení, které se mj. skládá i z detekce pozičních značek, lze vidět na obrázku 6.3.



Obrázek 6.3: Graf závislosti doby čtení QR kódu na velikosti obrazu

Jelikož proces čtení QR kódu se skládá z více kroků, byla provedena analýza náročnosti jednotlivých kroků z hlediska procesorového času (obrázek 6.4). Výsledku může být využito v budoucnu pro návrh případných optimalizací.



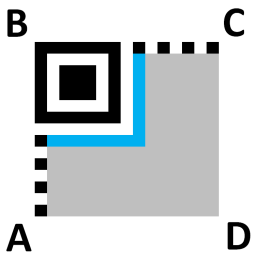
Obrázek 6.4: Graf procesorového času stráveného v jednotlivých krocích čtení QR kódu

Bylo zjištěno, že převážná část procesorového času je spotřebována právě v kroku detekce pozičních značek. Následně po ní se ukázal být náročný krok transformace QR kódu do normalizovaného zobrazení. Avšak ne samotná transformace, ale binarizace obrazu, která je prováděna ihned po transformaci v témže kroku. Překvapivě dobře dopadl krok přípravy obrazu, v němž je převáděn komprimovaný obraz v JPEG formátu na obraz ve stupních šedi. Další kroky čtení, které nejsou v kruhovém grafu uvedeny, ukrojily z celkového času pouhých 0,1%.

6.3 Možnosti budoucího vývoje

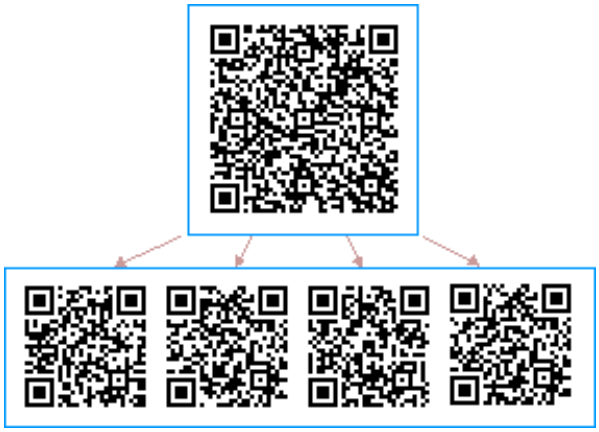
Velkým prostorem pro samotný vývoj se skýtá v C++ knihovně pro detekci a čtení QR kódů. Knihovnu by bylo možné rozšířit např. o podporu čtení Micro QR kódů, strukturovaného režimu nebo kódů ze zakřivených povrchů.

Přidání podpory pro čtení skupiny Micro QR kódů by znamenalo pouze změnit koncepci nacházení oblasti QR kódu. Ta by mohla být založena na sledování synchronizačních vzorů vedoucích z detekované poziční značky Micro QR kódu (obrázek 6.5). Pokud by sledování dosáhlo konce, tzn. strany QR kódu, byly by zde stanoveny rohy A a C. Tyto rohy by se staly zároveň i počátečními body pro vzorkování a nalezení čtvrtého rohu QR kódu D – viz kapitola 4.6.



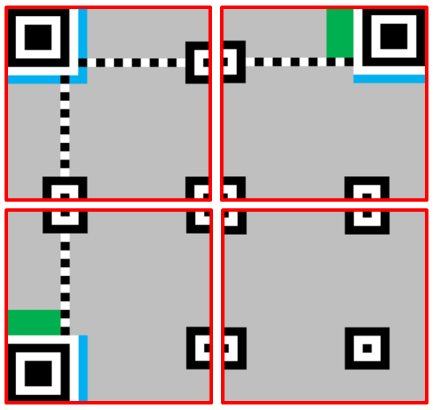
Obrázek 6.5: Micro QR kód

Podpora strukturovaného režimu (kapitola 2.3.2) nebyla pozorována na žádné z testovaných veřejně dostupných čteček pro platformu Android. Režim slouží pro umístění dat do více QR kódů (obrázek 6.6). Využívá se spíše v aplikačně uzavřených doménách např. pro značení výrobků s nedostatečným prostorem pro tisk v obou dimenzích. Hlavním problémem, který se zde vyskytuje, je rozlišení pozičních značek a přiřazení k jednotlivým QR kódům. Implementačně by se to dalo řešit opět sledováním synchronizačních vzorů vedoucích z jednotlivých pozičních značek. Pokud by ke značce vedly dva synchronizační vzory, byla by tato značka prohlášena jako levá horní. Aby mohlo být rozhodnuto, že v daném směru existuje synchronizační vzor, musel by být každý takový vzor zakončen sekvencí tmavých a světlých modulů poziční značky v poměru 1:1:3:1:1.



Obrázek 6.6: Rozdělení dat do více QR kódů pomocí strukturovaného režimu [9]

Čtení QR kódů ze zakřivených povrchů by šlo realizovat pomocí zarovnávacích značek v QR kódu. Samotný princip čtení je dokonce uveden v referenčním algoritmu pro čtení QR kódů v ISO normě. Podstata by spočívala v nalezení zarovnávacích značek a rozdělení QR kódu na dvourozměrné menší bloky (obrázek 6.7), které by poté byly samostatně převedeny perspektivní inverzní transformací do normalizovaného zobrazení. Po převodu všech bloků by se obraz sjednotil a mohla by být aplikována standardní metodika čtení.



Obrázek 6.7: Rozdělení QR kódu do bloků podle zarovnávacích značek

Rozšíření aplikace pro Android by bylo také možné. Zahrnovalo by však spíše vylepšení z pohledu uživatele, nikoliv však funkční z pohledu spolehlivosti a podpory čtení QR kódů. Hlavním přínosem by byla implementace souborového správce, který by byl využit pro správu QR kódů, obrázků a instalovaní zásuvných doplňků. Nyní je za tímto účelem nutno mít nainstalovanou aplikaci OI File Manager. Procházení složek a souborů je již v aplikaci implementováno, ale není doplněno o akce správy (mazání, přesouvání apod.). Druhým, implementačně méně náročným vylepšením, by mohlo být umožnění dekódování QR kódu z fotografie z jakéhokoliv souborového správce.

Kapitola 7

Závěr

Cílem této bakalářské práce bylo zanalyzovat problematiku dvourozměrných čárových QR kódů. Na základě získaných teoretických znalostí navrhnout algoritmy pro jejich automatické rozpoznávání a přečtení ve snímaném obraze. Dále tyto algoritmy implementovat a využít v aplikaci čtečky QR kódů pro mobilní platformu Android.

Během návrhu metodiky čtení QR kódů práce vychází z aktuální normy spravující standard QR kódů. Pro některé kroky čtení, které norma nespecifikovala, nebo nebyly pro naše účely příliš vhodné, bylo nutno prezentovat vlastní řešení. Jedná se zejména o přípravu obrazu pro strojové zpracování a lokalizaci QR kódů v obraze.

V rámci práce se podařilo implementovat všechny navržené postupy uvedené v kapitole 4. Automatické rozpoznávání QR kódů v obraze probíhá nad binarizovaným obrazem (kapitola 4.3) a je prováděno pomocí hledání kontur (kapitola 4.5). Při hledání kontur jsou v obraze lokalizovány tzv. poziční značky, které se vyskytují ve třech rozích QR kódu. Po lokalizaci těchto značek a určení čtvrtého rohu QR kódu (kapitola 4.6) dochází k inverzní perspektivní transformaci (kapitola 4.4) QR kódu do normalizovaného zobrazení. Samotného přečtení QR kódu je docíleno přiložením čtecí mřížky a vzorkováním bodů uvnitř buněk mřížky (kapitola 4.7).

V praktické části práce byla v jazyce C++ s využitím knihovny OpenCV vytvořena knihovna (kapitola 5.1), která zajišťuje lokalizaci a čtení QR kódů v obraze. Knihovna je napsána přenositelně a byla úspěšně otestována na mobilních architekturách ARMv6 a ARMv7a a desktopových architekturách x86 a x64.

Druhým produktem praktické části práce je aplikace čtečky QR kódů napsaná v jazyce Java (kapitola 5.2), jejíž funkčnost je závislá na dostupnosti předešlé knihovny. Samotná knihovna je potom v aplikaci čtečky QR kódů dynamicky načítána. Vzájemná komunikace mezi aplikací a knihovnou probíhá skrze nativní rozhraní JNI. Aplikace je unikátní od ostatních čteček QR kódů zejména svojí snadnou rozšiřitelností v podobě zásuvných doplňků, dokonce i za běhu aplikace.

Za účelem zhodnocení spolehlivosti a výkonnosti implementovaných funkcionalit, byla aplikace podrobena sadě testů (kapitoly 6.1 a 6.2), které prokázaly, že aplikace je použitelná v praxi, ale není vhodná pro čtení QR kódů z reálného obrazu. Detekce QR kódů v „běžně“ strukturovaném reálném obraze, jehož rozlišení odpovídalo rozlišení displeje, dosahovala na testovaných mobilních zařízeních frekvence obnovy 5-6 rámců za sekundu. Doba celého procesu čtení závisela na zvoleném rozlišení vstupního obrazu a počtu opravných kroků čtení, které se v případě selhání předešlého kroku mohou uplatnit. V porovnání s ostatními veřejně dostupnými čtečkami dosahovala aplikace v testech lepších výsledků, pokud QR kód obsahoval poškozené zarovnávací značky.

Budoucí vývoj práce (kapitola 6.3) by mohl směřovat zejména k optimalizaci a zvýšení přesnosti lokalizace QR kódů. Mohla by být implementována i podpora čtení více QR kódů v obraze a nové skupiny QR kódů tzv. Micro QR kódů. Vhodné by rovněž bylo aplikovat schopnost čtení ze zakřivených povrchů.

Literatura

[1] *About 2D Code : Bar code to 2D Code* [online]. [cit. 2012-04-17]. Dostupné z: <http://www.denso-wave.com/qrcode/aboutqr-e.html>.

[2] *Adaptive Thresholding* [online]. [cit. 2012-04-25]. Dostupné z: <http://homepages.inf.ed.ac.uk/rbf/HIPR2/adpthrsh.htm>.

[3] *Application Fundamentals* [online]. [cit. 2012-04-29]. Dostupné z: <http://developer.android.com/guide/topics/fundamentals.html>.

[4] *Barcode Contents* [online]. [cit. 2012-01-11]. Dostupné z: <http://code.google.com/p/zxing/wiki/BarcodeContents>.

[5] *Barcode Scanner* [online]. [cit. 2012-05-11]. Dostupné z: <https://play.google.com/store/apps/details?id=com.google.zxing.client.android>.

[6] *Barcode Symbologies* [online]. [cit. 2012-04-25]. Dostupné z: <http://www.neodynamic.com/Products/BarcodeSysmbologies.aspx>.

[7] *Micro QR Code* [online]. [cit. 2012-04-12]. Dostupné z: <http://www.denso-wave.com/qrcode/microqr-e.html>.

[8] *Online Barcode Generator* [online]. [cit. 2012-04-24]. Dostupné z: <http://www.terryburton.co.uk/barcodewriter/generator/>.

[9] *QR Code Features* [online]. [cit. 2012-05-12]. Dostupné z: <http://www.denso-wave.com/qrcode/qrfeature-e.html>.

[10] *QR Code Standardization* [online]. [cit. 2012-04-12]. Dostupné z: <http://www.denso-wave.com/qrcode/qrstandard-e.html>.

[11] *QR Reader for Android* [online]. [cit. 2012-05-11]. Dostupné z: <https://play.google.com/store/apps/details?id=uk.tapmedia.qrreader>.

[12] *Reed-Solomon* [online]. [cit. 2012-05-08]. Dostupné z: <http://sourceforge.jp/projects/reedsolomon/>.

[13] *ScanLife Barcode & QR Reader* [online]. [cit. 2012-05-11]. Dostupné z: <https://play.google.com/store/apps/details?id=com.ScanLife>.

[14] *Standards: Stacked Symbologies* [online]. [cit. 2012-04-25]. Dostupné z: <http://www.aimglobal.org/standards/stackedsymbologies.asp>.

[15] *Synchronization with Native Applications : Common items* [online]. [cit. 2012-01-11]. Dostupné z: <http://www.nttdocomo.co.jp/english/service/developer/make/content/barcode/function/application/common/index.html>.

[16] *What is a bar code scanner?* [online]. [cit. 2012-04-24]. Dostupné z: <http://www.denso-wave.com/en/adcd/fundamental/barcode/scanner.html>.

[17] *What is Android?* [online]. [cit. 2012-04-29]. Dostupné z: <http://developer.android.com/guide/basics/what-is-android.html>.

[18] *ISO/IEC 18004 : Information technology – Automatic identification and data capture techniques – QR Code 2005 bar code symbology specification*. Switzerland: ISO/IEC, 2006.

[19] ADAMS, R. *2-Dimensional Bar Code Page : Specifications For Popular 2D Bar Codes* [online]. [cit. 2012-04-25]. Dostupné z: <http://www.adams1.com/stack.html>.

[20] BENEDA, M. Homografie a epipolární geometrie. *Trilobit* [online]. Vydáno: 31.10.2010 [cit. 2012-04-29]. Dostupné z: <http://trilobit.fai.utb.cz/homografie-a-epipolarni-geometrie>.

[21] ČÍKA, P. Protichybové zabezpečení BCH kódem. *Elektrorevue* [online]. Vydáno: 13.3.2006 [cit. 2012-04-28]. Dostupné z: <http://www.elektrorevue.cz/clanky/06015/index.html>. ISSN 1213-1539.

[22] DAWSON, A. Getting the Most Out of QR Codes Using URI Schemes. *Six Revisions* [online]. Vydáno: 20.2.2012 [cit. 2012-04-28]. Dostupné z: <http://sixrevisions.com/web-development/qr-codes-uri-schemes/>.

[23] HALL, J. I. *Notes on Coding Theory* [online]. [cit. 2012-05-01]. s. 105. Dostupné z: <http://www.mth.msu.edu/~jhall/classes/codenotes/Cyclic.pdf>.

[24] KOTON, J.; ČÍKA, P.; KŘIVÁNEK, V. Samoopravné Reed-Solomonovy kódy. *Access server* [online]. Vydáno: 11.10.2006 [cit. 2012-04-12]. Dostupné z: <http://access.feld.cvut.cz/view.php?cisloclanku=2006080002>.

[25] MEIER, R. *Professional Android^{TM} 2 Application Development*. Indianapolis: Wiley Publishing, Inc., 2010. ISBN 978-0-470-56552-0.

[26] SKLAR, B. Reed-Solomon Codes. *Journal of the Society for Industrial and Applied Mathematics*. 1960, vol. 1, no. 3, s. 646–656. ISSN 1462-0332.

[27] STOCKTON, R.; SUKTANKAR, R.; MULLIN, M. Smarter Presentations: Exploiting Homography in Camera-Projector Systems. *Proceedings of International Conference on Computer Vision*. 2001.

[28] SUZUKI, S.; ABE, K. Topological structural analysis of digitized binary images by border following. *Computer Vision Graphics and Image Processing*. 1985, vol. 30, no. 1, s. 32–46. ISSN 0734-189X.

[29] ŠMEJKAL, L. Snímače čárových kódů : přehled trhu. *Identifikační systémy*. 2007, č. 11, s. 720–730.

Přílohy

Seznam příloh

Příloha A

Obsah DVD

Na elektronický nosič byly přiloženy následující součásti bakalářské práce:

- **BarcodesLibrary/** – C++ knihovna pro detekci a dekódování QR kódů;
  - **src/** – zdrojové kódy knihovny;
  - **doc/** – programová dokumentace knihovny v HTML formátu;
  - **lib/** – výstupní složka pro sestavenou knihovnu;
  - **tests/** – složka obsahující zdrojové kódy pro sestavení testů knihovny;
- **Doc/** – technická zpráva;
  - **zprava.pdf** – technická zpráva ve formátu PDF;
- **DocomoDecoderPlugin/** – plugin obsahující dekódér specifické aplikační vrstvy, který lze doinstalovat do aplikace čtečky QR kódů;
  - **src/** – zdrojové kódy pluginu;
  - **DocomoDecoderPlugin.jar** – plugin ve formě archivu JAR, obsahující třídy zkompilované pro Dalvik VM;
- **GNUMake_OSSLib/** – GNU Make knihovna s makry pro překlad knihovny BarcodesLibrary;
- **OpenCV/** – sestavená OpenCV knihovna pro cílové mobilní architektury procesorů ARMv6 a ARMv7a;
- **QRReader/** – aplikace čtečky QR kódů pro Android;
  - **src/** – zdrojové kódy aplikace;
  - **doc/** – programová dokumentace aplikace v HTML formátu;
  - **QRReader.apk** – sestavená výsledná aplikace čtečky QR kódů pro Android.

Příloha B

Ukázka výpočtu bitů korekce pro metainformace QR kódu

V blocích metainformací QR kódu jsou vždy přidávány za samotné informační bity i bity korekce chyby. Pro výpočet těchto bitů se využívá cyklických Bose-Chaudhuri-Hocquenghem kódů, zkráceně BCH. Redundantní bity jsou tvořeny generujícími polynomy $G(x)$, jejichž tvar je pevně stanoven. Pro kódování metainformace formátu QR kódu je konkrétně využito kódu BCH(15, 5)[1] a pro kódování metainformace verze QR kódu BCH(18, 6). [18]

Generující polynom formátu QR kódu:

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

Generující polynom verze QR kódu:

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

Zabezpečení informačních bitů probíhá s využitím vztahu

$$\frac{f(x)x^{n-k}}{G(x)} = h(x) + \frac{r(x)}{g(x)}, \tag{B.3}$$

kde redundantní bity jsou zjištěny jako zbytek po dělení generujícím polynomem $G(x)$. [21]

Postup výpočtu redundatních bitů verze QR kódu

– vstupní informační bity: 000111 (verze 7)

1. Převedení binárního řetězce informačních bitů do polynomického tvaru:

$$000111 \Rightarrow x^2 + x + 1. \tag{B.4}$$

2. Zvýšení mocnin polynomu v závislosti na BCH(n, k) o $n-k$ pozic, neboli vynásobení polynomem x^{n-k}, zde BCH(18, 6):

$$x^2 + x + 1 \Rightarrow x^{14} + x^{13} + x^{12}. \tag{B.5}$$

[1] BCH(n, k) – Zápis pro Bose-Chaudhuri-Hocquenghem kódy, kde n udává, na kolika bitech bude výsledný řetězec bloku metainformace (včetně redundantních bitů) a k, pro kolik informačních bitů je korekce chyby počítána.

3. Vydělení vzniklého polynomu generujícím polynomem:

$$\frac{x^{14} + x^{13} + x^{12}}{G(X)} = x^2 + \frac{x^{11} + x^{10} + x^7 + x^4 + x^2}{x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^5 + x^2 + 1}. \tag{B.6}$$

4. Převod zbytku po dělení na binární řetězec (redundatních bitů):

$$x^{11} + x^{10} + x^7 + x^4 + x^2 \Rightarrow 110010010100. \tag{B.7}$$

5. Konkatenace informačních bitů a jejich redundatních bitů korekce chyby:

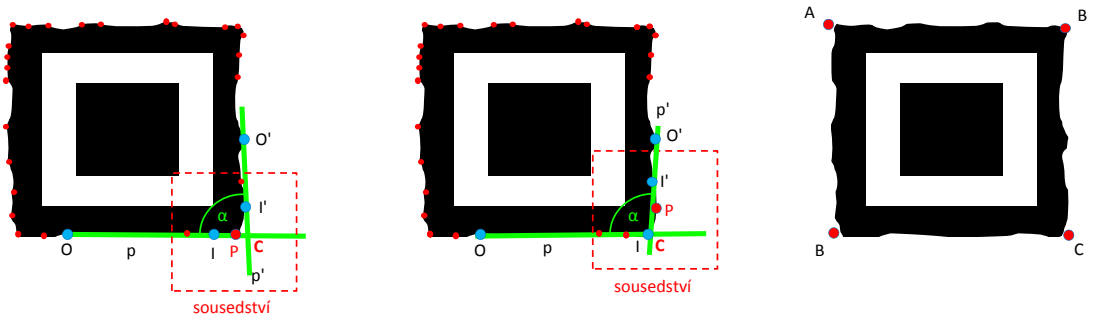
$$000111 + 110010010100 \Rightarrow 000111110010010100. \tag{B.8}$$

6. Aplikace XOR masky 101010000010010, jedná-li se o kódování metainforamce formátu QR kódu, k zamezení vzniku binárního řetězce s nulovými bity ve všech pozicích.

Příloha C

Algoritmus hledání rohů polygonu

Algoritmus hledání rohů polynomu byl navržen pro hledání čtyř bodů (rohů) poziční značky, která byla definovaná množinou bodů, jež ohraničovaly danou značku. Obecně lze ovšem algoritmus použít pro jakýkoliv polynom, princip jeho činnosti můžeme vidět na obrázku C.1.



Obrázek C.1: Zobrazení dvou kroků algoritmu a výsledku hledání rohů polynomu

Vstupními parametry algoritmu jsou:

- velikost sousedství v pixelech V,
- interval povolených úhlů R, jež mohou svírat přímky p a p', aby jejich průsečík byl uznán jako roh polygonu.

Slovní popis algoritmu

1. Pokud množina bodů kontur obsahuje méně než tři body, skonči.
2. Pro každý bod kontury vytvoř strukturu, která obsahuje body I, O, I' a O'.

   (a) Obsahuje-li sousedství právě zpracovávaného bodu ve směru hodinových ručiček alespoň jeden bod, ulož do I právě ten nejbližší v daném směru, jinak ulož do I právě zpracovávaný bod. Totéž proveď i ve směru proti směru hodinových ručiček a nastav bod I'.

   (b) Existuje-li mimo sousedství právě zpracovávaného bodu alespoň jeden bod, ulož do O nejbližší bod mimo sousedství ve směru hodinových ručiček. Neexistuje-li

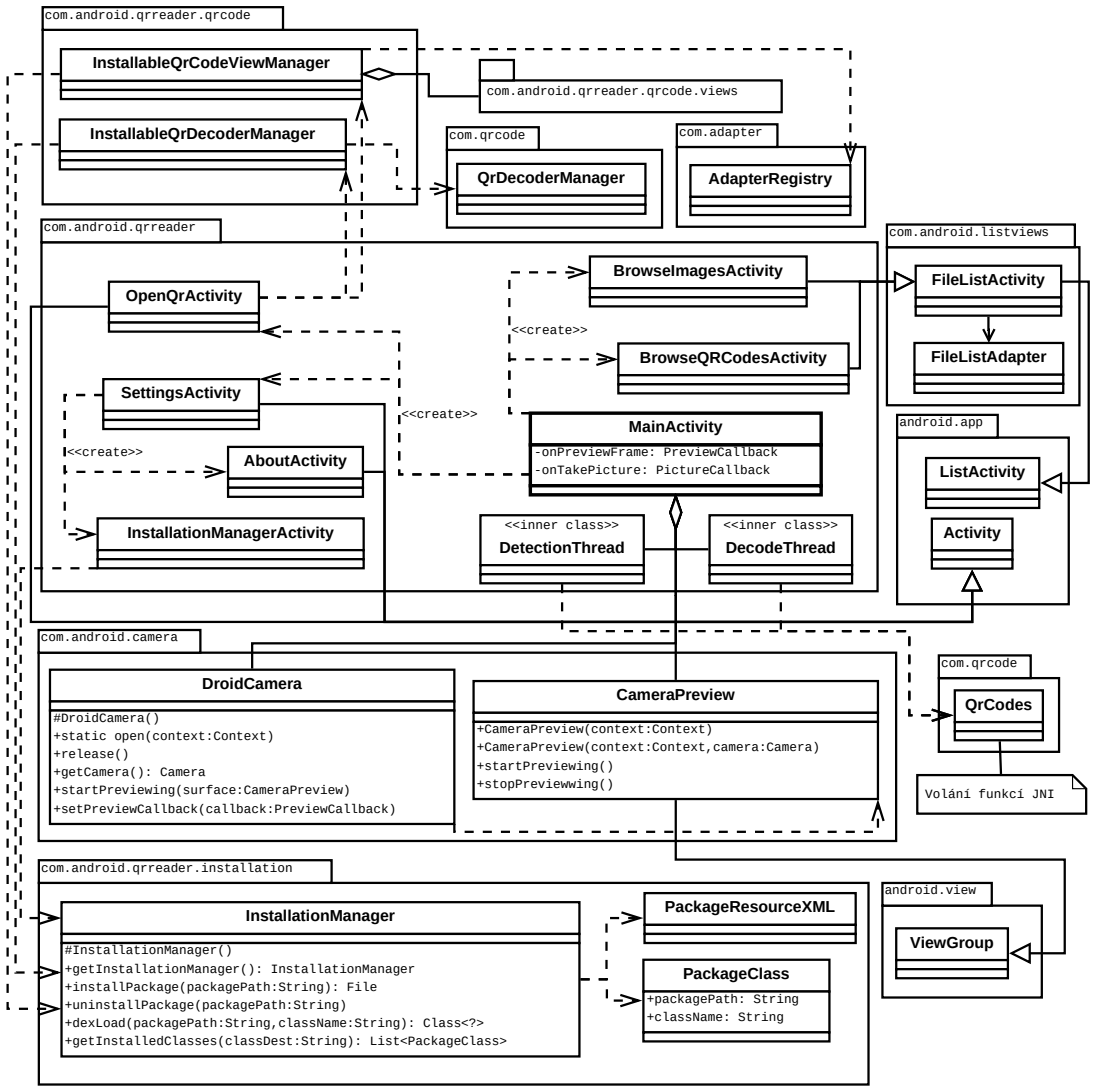
mimo sousedství právě zpracovávaného bodu žádný bod a v I není nastaven právě zpracovávaný bod, ulož do O právě zpracovávaný bod. Neexistuje-li mimo sousedství právě zpracovávaného bodu žádný bod a v I je nastaven právě zpracovávaný bod, ulož do O nejbližší bod ve směru hodinových ručiček od právě zpracovávaného bodu. Totéž proveď i ve směru proti směru hodinových ručiček a nastav bod O' s ohledem na I'.

3. Projdi všechny vytvořené struktury a sestroj z bodů I a O přímku p a z bodů I' a O' přímku p'. Zjisti úhel α, jež svírají obě dvě přímky a v případě, že $\alpha \in R$, vlož průsečík těchto přímek do seznamu získaných rohů S.

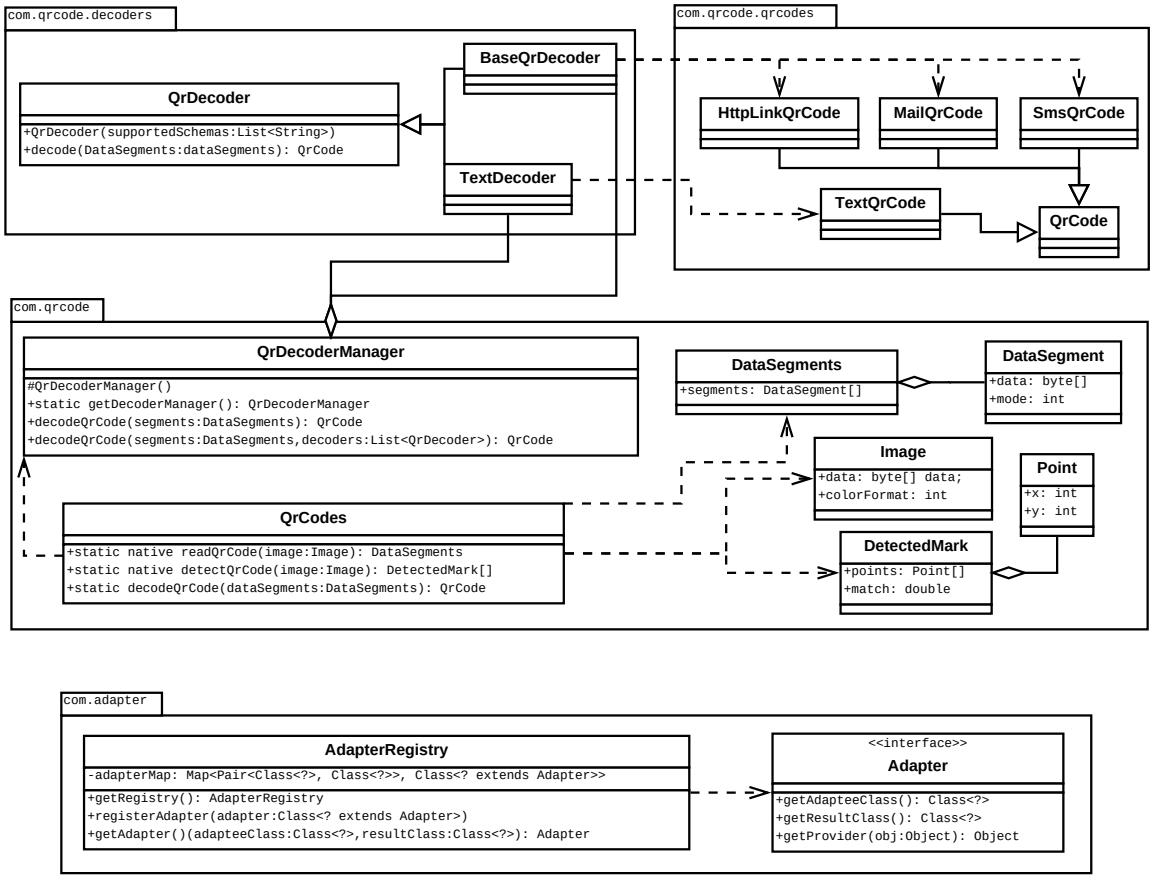
4. Projdi všechny získané rohy S a ověř, zda se v sousedství rohů nenachází jiný z nalezených rohů S. Pokud je nalezen v sousedství nějakého rohu jiný roh, odstraň oba dva rohy a vlož do seznamu rohů nový roh daný aritmetickým průměrem souřadnic obou rohů a opakuj krok 4, dokud sousedství každého rohu není prázdné.

Příloha D

Diagram tříd aplikace



Obrázek D.1: Zjednodušený diagram tříd aplikace čtečky QR kódů – 1. část



Obrázek D.2: Zjednodušený diagram tříd aplikace čtečky QR kódů – 2. část