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

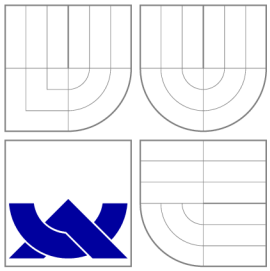
DETEKTOR UNIFORM MARKER FIELDS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

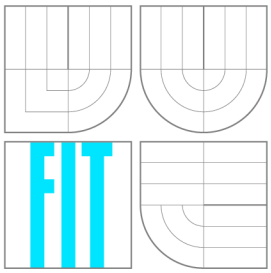
AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ MAŠEK

BRNO 2014



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

DETEKTOR UNIFORM MARKER FIELDS

DETECTOR UNIFORM MARKER FIELDS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ MAŠEK

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. ADAM HEROUT, Ph.D.

BRNO 2014

Abstrakt

Tato práce pojednává o detekci Uniform Marker Fields a umístění kamery v prostoru. Jsou zde popsány vlastní kroky detekce UMF, platforma Windows Phone 8, práce s DirectX a také pojem rozšířená realita. V textu je popsána zároveň implementace a návrh demo aplikace a celá architektura projektu. Výsledkem této práce je aplikace využívající detektor UMF a vykreslující 3D objekt do scény. Na závěr je aplikace otestována a zhodnocena.

Abstract

This thesis deals with the detection of Uniform Marker Fields and the position of a camera in a space. The steps of the UMF detection, the Windows Phone 8 platform, DirectX working and the concept of augmented reality are described in the thesis. Implementation and design of the demo application together with the whole architecture of the project is described in the thesis. The result of the thesis is an application using the UMF detector and plotting a 3D object into a scene. Finally the application is tested and evaluated.

Klíčová slova

Uniform Marker Fields, UMF, Rozšířená realita, Marker, Windows Phone 8, WP8, Direct3D, PnP

Keywords

Uniform Marker Fields, UMF, Augmented reality, Marker, Windows Phone 8, WP8, Direct3D, PnP

Citace

Jiří Mašek: Detektor Uniform Marker Fields, diplomová práce, Brno, FIT VUT v Brně, 2014

Detektor Uniform Marker Fields

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana doc. Ing. Adama Herouta Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Mašek
25. května 2014

Poděkování

Rád bych poděkoval vedoucímu práce doc. Ing. Adamu Heroutovi Ph.D. za odborné vedení. Také velice děkuji panu Ing. Istvánu Szentandrási za pomoc při řešení problémů s algoritmem. Poděkování patří i mé rodině za podporu a přítelkyni za zapůjčení testovacího zařízení.

Obsah

Kapitola 1

Úvod

V dnešní době se objevují aplikace, které umožňují svět okolo nás vylepšit či rozšířit za pomoci digitálních technologií. Tomuto vylepšení se říká rozšířená realita, která se stává velmi populární. V tomto odvětví jsou využívány poslední technologické novinky a to jak v oblasti hardwaru, tak i softwaru.

Rozšířená realita pro svůj správný chod potřebuje dostatek informací z okolního reálného světa. Dosud nebyl objeven žádný způsob, jak by bylo možné tyto informace získat z prosté scény snímané kamerou a proto se využívají objekty se známými vlatnostmi. Z tohoto důvodu je rozšířená realita technologie, která má velké nároky na výkon a paměť zařízení, na kterém je provozována. To je jeden z hlavních důvodů, proč na trhu s aplikacemi na mobilní platformy podobných aplikací není mnoho. Dnešní moderní mobilní zařízení však mají stále větší výkon, což umožňuje vytvářet aplikace s většími nároky, jako jsou aplikace s rozšířenou realitou. Těmto aplikacím také nahrává alogirtmus pro detekci Uniform Marker Fields (dále jen UMF), který byl vytvořen tak, aby nebyl náročný na výkon a paměť zařízení a přesto dosahoval dobrých výsledků. Tato práce vychází z článku [7], který pojednává právě o UMF a jeho detekci a zabývá se implementací detektoru UMF na mobilní platformě Windows Phone 8. Cílem je také zjistit, zdali dnešní zařízení s Windows Phone 8 jsou schopna provozovat tento typ aplikací.

V této práci, v kapitole 1 podrobněji popíšeme rozšířenou realitu a především UMF a jeho detekce. V kapitole 2 se budeme věnovat algoritmu UMF, jeho principu a využití. V následujícím textu (kapitola 3) nejprve popíšeme mobilní platformu Windows Phone 8. Především se zde zaměříme na hardware mobilních zařízení, vývoj v nativní kódu a na DirectX. V následující kapitole 4 popisujeme návrh aplikace, její architekturu a implementaci jednotlivých části algoritmu. Současně zde také popisujeme funkce demo aplikace. Následuje kapitola 5, ve které se zabýváme testováním algoritmu za různých podmínek a na různých zařízeních a zde také zobrazujeme výsledky testovaní. V závěru 6 shrnujeme celou práci, zkušenosti s vývojem, možnosti vylepšení a pokračování projektu.

Kapitola 2

Rozšířená realita

Pojem rozšířená realita se občas nesprávně zaměňuje s virtuální realitou. Virtuální realita je počítačem vytvořená realita, se kterou lze různymi způsoby interagovat, ale s reálným světem nesouvisí. Rozšířená realita naopak s reálným světem koexistuje. Reálný svět je za pomocí této technologie rozšiřován a obohacován. Jsou přidávány různé počítačem generované dodatečné informace, 3D/2D objekty, obrázky, videa a další prvky. Tímto způsobem jsou uživateli předávány informace, které by pro něj jinak byly velmi složitě dosažitelné.

Každý systém virtuální reality splňuje následující obecné charakteristiky[3]:

1. Kombinuje reálný svět s virtuálním
2. Reaguje v reálném čase
3. Pracuje ve 3D prostoru

Pro správnou funkčnost systému rozšířené reality je nutné určit pozici a orientaci kamery vzhledem k reálnému světu. Existují různé techniky pro dosažení tohoto cíle:

- Technika využívající senzory
- Technika založená na zpracování obrazu z kamery
- Hybridní technika využívající oba přístupy

V prvním případě se jako senzor nejčastěji využívá GPS a elektronický kompas. Díky GPS jsme schopni zjistit umístění zařízení v prostoru (na světě) a pomoci elektronického kompasu určíme, kterým směrem se díváme, tedy orientaci v prostoru. Poté je možné rozšířit reálný svět o dodatečné informace či objekty. Druhá technika je náročnější na výkon a paměť zařízení, na kterém je provozována. Obraz musí projít několika složitými a náročnými algoritmy, abychom z něho získali potřebné informace pro výpočet pozice a umístění kamery v prostoru. Třetí způsob je kombinací již zmíněných dvou technik.

Největší aktivita ve výzkumu se soustředí právě do techniky založené na zpracování obrazu z kamery. Využívají se znalosti z mnoha oblastí jako počítačového vidění či zpracování obrazu. V této práci se budeme zabývat právě touto technikou.

2.1 Markery

Abychom mohli určit správnou a přesnou pozici a orientaci kamery, je nezbytné, abychom v obraze nalezli předem definované prvky. Těmito prvky mohou být:

- Markerless - prvky, které se vyskytují v reálném světě a jsme schopni je rozeznat. Jsou to významné body nebo oblasti v obraze jako rohy, hrany a jiné.
- Marker - prvky, které jsou do reálného světa přidány uměle. Těmto prvkům se říká markery a jejich vlastnosti jsou předem známé a lze je rychle a efektivně detekovat.

Pojem marker si lze představit jako dvourozměrný vzor, který se vkládá do snímané scény. Takový marker může mít více účelů. Jedním z nich je přenos informace. Typickým příkladem tohoto typu markerů jsou QR kódy, či čárové kódy. Dalším účelem je lokalizace v obraze. Právě takové typy markerů se často využívají v systémech s rozšířenou realitou [5].

Marker se často skládá z jednoduchých geometrických tvarů jako čtverců, obdélníků apod. Tyto tvary bývají většinou černobílé, jelikož detekce probíhá nejčastěji prahováním, a proto je vhodné mít co největší rozdíl v jasu, aby se minimalizoval vliv osvětlení na detekci markeru [10].

Systém s rozšířenou realitou, který nepoužívá pro určení pozice v prostoru markery, hledá ve scéně jiné objekty, na jejichž základě se vypočítá pozice a orientace kamery v prostoru. Bohužel zatím neexistuje algoritmus, který by v reálné scéně nalezl dostatek požadovaných prvků, díky kterým lze určit pozici a orientaci kamery v prostoru.

Při použití markerů jsme schopni mnohem efektivněji nalézt požadované prvky ve scéně tak, abychom mohli určit pozici a orientaci kamery v prostoru. A to především proto, že marker je navrhován s cílem efektivní detekce a vlastnosti markeru jsou předem definované a proto lépe detekovatelné.

V této práci se budeme zabývat právě detekcí markerů v obraze reálného světa snímaným kamerou. Příklady aplikací se systémem rozšířené reality využívající markery pro platformu Windows Phone 8 jsou uvedeny později.

2.2 Existující aplikace

Aplikace pro systém rozšířené reality na mobilní platformu Windows Phone 8 jsou vzácností. Jednak je Windows Phone 8 poměrně mladá platforma a množství aplikací v marketu nedosahuje počty na marketu Androidu či iOSu. Ale i tak existuje pár toolkitů, za jejichž pomoci lze vytvořit aplikaci se systémem rozšířené reality.

2.2.1 Silverlight ARToolkit

Prvním z nich je Silverlight ARToolkit [11]. ARToolkit je poměrně známý, jelikož existuje na více platformách včetně Androidu. Pro zjištění pozice a orientace používá marker.

Tento marker je dvoubarevný (černobílý) a vyznačuje se čtvercovým černým okrajem, v němž je obsažen libovolný obrázek odlišný od ostatních.

Postup detekce je následovný [11]:

- Obraz je prahováním převeden na binární obraz.
- V binárním obraze jsou nalezeny všechny čtverce.

Obrázek 2.1: Marker pro ARToolkit [11].

- Ve všech nalezených čtvercích se porovnávají vzory se vzory v databázi. Pokud nějaký vzor odpovídá, je čtverec detekován jako marker.
- Na základě markeru v databázi a detekovaného markeru je vypočítána pozice a orientace kamery v prostoru.

ARToolkit má svá omezení. Největším je skutečnost, že marker musí být celý v zorném poli kamery a nesmí být překryt žádným jiným objektem [11]. A toto omezení sebou přináší další. Například pohyb kamery v prostoru. Ten je v tomto případě možný jen když použijeme více markerů naskládaných vedle sebe, a i tak jsou výsledky velmi špatné, jelikož při přechodu z jednoho markeru na další musíme zajistit, aby byl minimálně jeden zcela v zorném poli kamery. Problémem je také prahovaní obrázku. Při nevhodném osvětlení markeru dochází ke špatnému prahování a marker není detekován vůbec nebo velmi špatně. Marker tohoto typu obsahuje dvě části. První je silný černý okraj markeru, za jehož pomoci detekujeme marker v obraze. A druhým je vzor uvnitř, kterým se marker identifikuje.

Obrázek 2.2: Překrytí markeru jiným objektem.

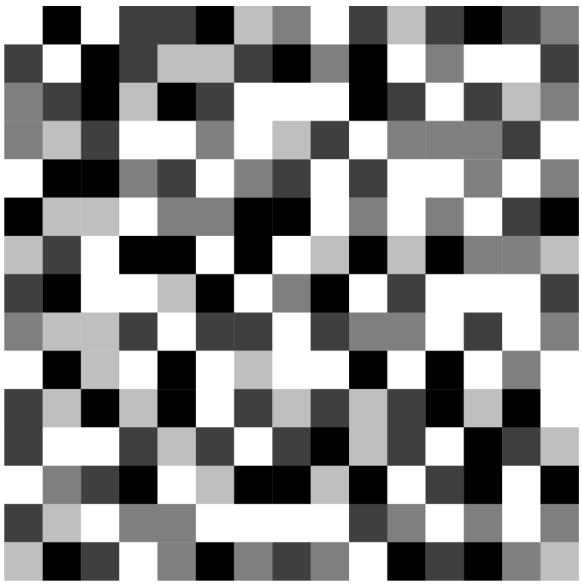
Na obrázku 2.2 můžeme vidět, že v momentě, kdy marker překryjeme například perem, obrázek již není vykreslen správně. Korektní vykreslení by bylo přímo nad markerem, který by nebyl poté vidět.

2.2.2 Geo AR Toolkit

GART (Geo AR Toolkit) je případ rozšířené reality, která pro určení pozice a orientace využívá senzorů v mobilním zařízení. Tento toolkit umožňuje rychlý a efektivní vývoj aplikací s rozšířenou realitou, které zobrazují dodatečné informace o prostředí, kde se pohybujeme, za pomoci senzorů mobilního zařízení [9].

2.3 Uniform Marker Fields

Uniform Marker Fields (dále jen UMF) je technologie, která byla navhrnuta s cílem překonat nedostatky dosavadních markerů.



Obrázek 2.3: Marker 15x15 o velikosti okna 3.

UMF si klade následující cíle [7]:

- Pokrytí velké plochy markerem a zároveň možnost detekce jen z malé části markeru.
- Odolnost vůči velkým pozorovacím úhlům.
- Odolnost vůči různým světelným podmínkám jako přímé světlo, stíny, různá intenzita světla.
- Odolnost vůči překrytí markeru jinými objekty
- Rychlá detekce markeru

UMF je postaveno na myšlence, že všechny hrany v obraze se protínají ve dvou hlavních bodech. Těmto bodům se říká úběžníky scény. Za pomoci těchto úběžníků a matematických formalismů jsme schopni detekovat celou mřížku markeru. Oproti markeru ARToolit jsou prvky pro identifikaci a detekci rovnoměrně rozprostřeny po celé ploše markeru, což vyjadřuje slovo Uniform v názvu UMF. UMF je složeno z jednotlivých modulů, které se navzájem překrývají, což představuje termín Marker Field. Každý modul je jedinečný v celém markeru ve všech možných rotacích. Hrany mezi čtverci jednoznačně identifikují jeden z modulů markeru. Díky tomu jsme schopni určit orientaci a pozici kamery v prostoru [7].

Původní návrh UMF byl pouze binární. Tedy černobílé čtverce. Avšak pro zlepšení detekce je použit vícebarevný marker. Konkrétně odstíny šedi, díky nimž je v obraze více hran a tím i více přímek [7].

2.4 Orientované okno

Orientované okno je základem UMF, poskládáním jednotlivých orientovaných oken (modulů) získáme dohromady celý marker. Každé orientované okno musí být unikátní v celém markeru a to ve všech možných směrech, abychom mohli určit pozici a orientaci orientovaného okna v markeru. Maximální počet orientovaných oken lze vypočítat na základě velikosti okna n a počtu barev k následovně [24]:

$$N \leq \frac{k^{n^2} - k\left[\frac{n^2+1}{2}\right]}{4} \tag{2.1}$$

Pokud chceme zjistit maximální velikost markeru, tedy šířku a výšku, můžeme tyto hodnoty spočítat následovně [24]:

$$(h - n + 1)(w - n + 1) \leq \frac{k^{n^2} - k\left[\frac{n^2+1}{2}\right]}{4} \tag{2.2}$$

Snímek markeru lze pořídit za různého osvětlení, mohou se však vyskytnout stíny a jiné problémy. Proto není vhodné při hledání umístění a orientace okna v markeru používat barvy čtverců. V UMF se používá porovnání směru gradientu. Tedy výsledkem porovnání je menší, větší či rovno (<,>,=). Díky tomu dokážeme odstranit problémy s osvětlením. Ale na druhou stranu zde dochází k obtížím, že i když budeme mít 5 barev (k=5), tak se může stát, že dostaneme dvě okna, jejichž porovnání gradientů budou totožná, přestože barvy jednotlivých čtverců budou odlišná. Příkladem je obrázek 2.4, kde barvy čtverců v oknech 2.4b a 2.4c jsou sice odlišné, ale při porovnání jejich směrů gradientů dojdeme ke stejnému výsledku. Díky tomu se zvyšují nároky na generátor markerů a taktéž se snižuje maximální velikost markeru. To proto, že při tomto stylu porovnávání máme vždy maximálně 3 výsledky (<,>,=) a tedy počet barev je vždy roven 3 ($k = 3$) [7].

2.5 Detekce UMF

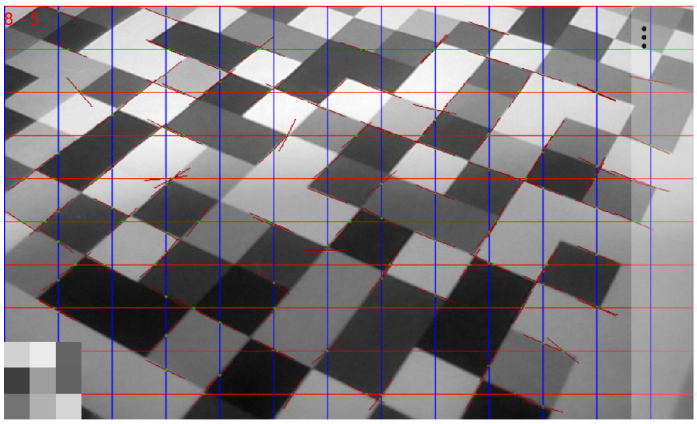
Následující text vychází z článku [7]. Detekce UMF je složena z několika kroků, které budou popsány později. Rozdíl oproti jiným detektorům (ARToolkit) je to, že detektor UMF nerozlišuje prvky pro lokalizaci markeru a prvky pro identifikaci markeru. Jiné detektory nejprve detekují, kde se marker v obraze nachází a až poté se provádí identifikace, což je mnohem náročnější.

(a) (b) (c)

Obrázek 2.4: 2.4a ukazuje porovnání směru gradientu hrany. 2.4b a 2.4c při porovnání směru gradientu hrany, jsou tyto okna totožná. Inspirováno [24].

2.5.1 Detekce hran

Prvním krokem je nalezení hran v obraze. V obraze se umístí vertikální a horizontální přímky, které jsou od sebe vzdáleny o určitý, předem nastavený offset. Těmto přímkám se říka *rozkladové řádky*. Výsledkem je mřížka z řídce rozmístěných přímek. Hrany se poté hledají právě na těchto horizontálních a vertikálních přímkách. Díky procházení pixelů pouze na těchto přímkách zpracováváme minimum pixelů z celého obrazu a tedy dosahujeme větších rychlostí při zpracování.

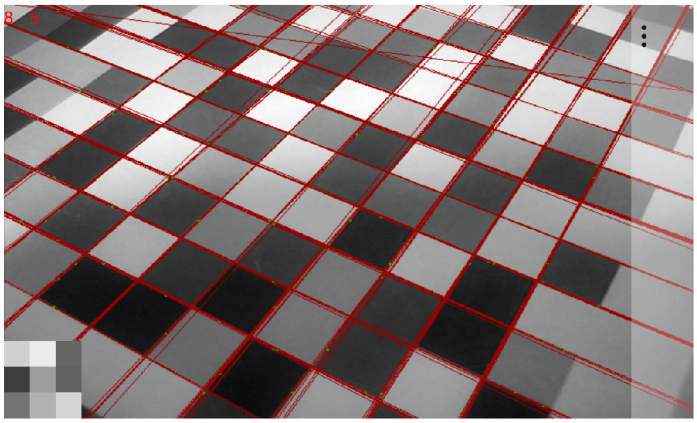


Obrázek 2.5: Rozkladové řádky - červené vertikální a modré horizontální přímky, Edgely - nalezené hrany a jejich směry určené pomocí Sobelova operátoru.

Vlastní hledání hran je prováděno pomocí algoritmu adaptivního prahování, které je založeno na plovoucím průměru o velikosti okna k. Po nalezení hran je nutné najít směr hrany. Toho je dosaženo použitím Sobelova operátoru, pomocí kterého dostaneme odhad směru gradientu. Normálový vektor odhadnutého směru gradientu nám určuje přibližný směr detekované hrany. Výsledný směr je však velmi hrubý a je třeba ho dále upřesnit, jinak by nebylo možné získat hlavní úběžníky scény. Výsledkem tohoto kroku je seznam hran a jejich směrů. Takovým hranám budeme říkat edgely [8]. Na obrázku 2.6 jsou zobrazeny červeně horizntální a modře vertikální rozkladové řádky.

2.5.2 Upřesnění směru přímek

Jak již bylo uvedeno, je nutné upřesnit hrubý směr odhadnutý Sobelovým operátorem. Upřesnění je provedeno posouváním se v odhadnutém směru a opětovném hledání edgelu. Potom se směr upřesní pomocí původního a nově nalezeného edgelu. Pokud však žádný edgel nalezen není, je původní edgel zahozen. Tento krok se opakuje několikrát po sobě a pokud není dosaženo stanoveného počtu iterací, je opět edgel zahozen. Toto upřesnění je prováděno v obou směrech. Tedy v odhadnutém směru hrany a v opačném směru odhadu. Tím dosáhneme lepšího upřesnění přímky. Na obrázku 2.6 je vidět výsledek tohoto upřesnění.

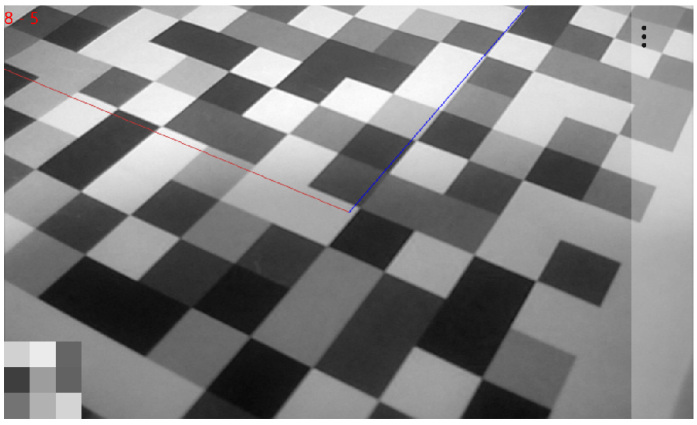


Obrázek 2.6: Upřesněné přímky - přímky, reprezentující upřesněné směry jednotlivých edgelů.

2.5.3 Nalezení úběžníků

Dalším krokem algoritmu je nalezení dvou hlavních úběžníku scény. Tyto úběžníky odpovídají průsečíkům nalezených přímek. Bohužel nalezené přímky nejsou zcela přesné, proto se jejich průsečík nenachází v jednom konkrétním bodě. Pro nalezení jednotlivých úběžníků je třeba přímky roztřídit do dvou hlavních skupin, kde každá skupina odpovídá právě jednomu hledanému úběžníku. Toto roztřídění je prováděno za pomoci metody RANSAC, která přímky rozdělí do dvou hlavních skupin a přímky, které nevyhovují ani jedné, jsou zahozeny. Z rovnic přímek vytvoříme matici M o velikost 3xN, kde N je počet přímek. Výsledná korelační matice C se vytvoří vynásobením matice M s transponovanou maticí M^T. Korelační matice je symetrická o velikosti 3x3. Díky tomu je možné nalézt hledaný úběžník velmi přesně a efektivně. Podrobněji bude tento krok popsán v kapitole 4.4.4.

Samotný úběžník je nalezen nasazením nadroviny přes všechny pozorované přímky [7]. Normála nadroviny je poté vypočtena pomocí eigen-dekompozice použitím korelační matice. Výsledkem eigen-dekompozice jsou vlastní hodnoty a vektory v homogenních souřadnicích. Hledaný úběžník je roven vlastnímu vektoru, který odpovídá nejmenší vypočtené vlastní hodnotě. Po vydělení vektoru homogenní souřadnicí dostáváme souřadnice hledaného úběžníku. Tento krok je proveden na obě skupiny přímek a výsledkem jsou dva hlavní úběžníky scény. Výsledek je vidět na obrázku 2.7



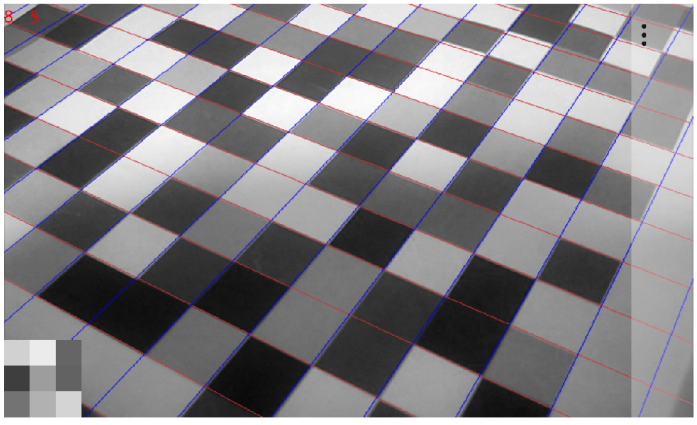
Obrázek 2.7: Úběžníky - přímky v obrázku zobrazují nalezené úběžníky.

2.5.4 Určení vějířů mřížky

Následujícím krokem je rekonstrukce mřížky. Pokud známe oba hlavní úběžníky scény, lze za jejich pomoci určit horizont h (přímka procházející oběma úběžníky). Poté lze libovolnou přímku mřížky vypočítat dle následující rovnice

$$bl_i = \hat{I}_{base} + (ki + q)\hat{h}, \tag{2.3}$$

kde $\hat{I}_{base}$ je libovolná přímka, která prochází úběžníkem a je jiná od horizontu. Ideální je přímka procházející středem obrazu a odpovídajícím úběžníkem. Hodnota (ki+q) se vypočítá pro všechny jednotlivé přímky. Potom se tyto přímky dle hodnoty (ki+q) shluknou a ke každému shluku je přiřazeno odpovídající i. Pak za pomoci lineární regrese je získána rovnice přímky f(i)=ki+q. Díky této rovnici jsme schopni určit libovolnou přímku, která reprezentuje hranu mřížky jen dosazením správného i.

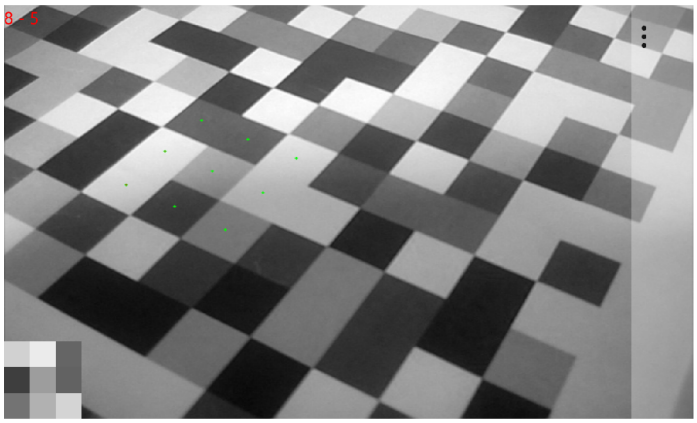


Obrázek 2.8: Určené vějíře mřížky.

2.5.5 Nalezení pozice a orientace v markeru

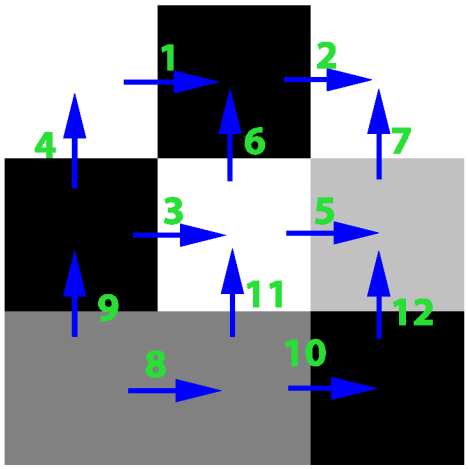
V této fázi jsme již schopni získat libovolnou přímku markeru pouhým dosazením i do rovnice 2.3. Po dosazení celého čísla za i dostaneme přesně přímku, která odpovídá jedné z hran v markeru. Pokud však dosadíme za i např. 0.5, dostaneme přímku, která prochází

přesně středem čtverce markeru. Tedy prochází mezi hranami markeru. Za tohoto předpokladu jsme schopni získat dvě přímky, kde každá odpovídá jednomu úběžníku a jejich průsečík je přesně bod uprostřed jednoho ze čtverců markeru. Tímto způsobem je možno získat všechny středy a jednotlivé barvy čtverců markeru.



Obrázek 2.9: Detekované okno a jeho středy čtverců.

Na základě barev čtverců markeru dochází k porovnávání, které bylo popsáno dříve 2.4 a jeho výsledkem je hodnota, která toto porovnání reprezentuje. Může to být řetězec „<>><<>=><<><“ nebo čísla „-111-1-1101-1-11-1“ a další způsoby reprezentace. Totéž porovnávání se provádí při načítání markeru a každé porovnání je uloženo do struktury. Touto strukturou může být hash tabulka, strom a další struktury. Zde se však kromě porovnání ukládá i pozice a orientace okna. Poté je výsledek porovnání barev detekovaných středů hledán v dané struktuře. Jakmile je dané okno nalezeno, získáme jeho pozici a orientaci, díky čemu jsme schopni určit pozici kamery v prostoru.



Obrázek 2.10: Ukázka pořadí porovnání barev. Každá šipka je jedno porovnání.

Kapitola 3

Windows Phone 8

Windows Phone 8 je nástupcem mobilní platformy Windows Phone 7. S jeho příchodem došlo k několika změnám oproti Windows Phone 7. Především bylo umožněno dávat do mobilních zařízení lepší hardware, a to jak po stránce procesoru a paměti, tak i displeje a dalších senzorů. Také byl umožněn vývoj v nativním kódu, tedy v C++, což je pro tuto práci klíčové. V moderních mobilních telefonech se začínají objevovat procesory, které podporují NEON technologii. Tato technologie umožňuje podstatné urychlení výpočtu, avšak vývoj přináší i svá negativa.

3.1 Minimální hardware v mobilním zařízení s Windows Phone 8

Windows Phone 8 stejně jako jeho předchůdce stanovuje pro výrobce telefonů s tímto operačním systémem minimální hardwarové požadavky. Windows Phone 8 přináší v tomto odvětví mírné pokroky oproti minulé verzi.

- Qualcomm Snapdragon S4 dual-core procesor
- Minimálně 512 MB RAM pro WVGA telefony a minimálně 1 GB RAM pro 720p a WXGA telefony
- Minimálně 4 GB flash memory
- GPS a A-GNSS; GLONASS je podporována, pokud se výrobce rozhodne tuto funkčnost přidat
- Podpora pro micro-USB 2.0
- 3.5mm stereo konektor s podporou 3-tlačítkové detekce
- Zadní AF kamera s LED nebo Xenon bleskem, volitelně přední kamera (obě musí být VGA nebo lepší) a speciální tlačítko pro kameru
- Akcelerometr, proximity a světelný senzor, samozřejmostí je vybračním motorek (magnetometer a gyroskop jsou volitlené)
- 802.11b/g a Bluetooth (802.11n je volitelný)

- Podpora DirectX s hardwarovou akcelerací pro Direct3D pomocí programovatelné GPU
- Multidotykový kapacitní displej s minimálně 4 současnými body dotyku

S touto minimální konfigurací můžeme počítat u každého mobilního zařízení s operačním systémem Windows Phone 8 [18]. Přináší to velkou výhodu při vývoji, jelikož nemusíme složitě řešit výkonnostní rozdíly mezi různými mobilními zařízeními a výrobci.

3.2 NEON technologie

Jak již bylo zmíněno, v dnešních moderních mobilních zařízeních se začínají objevovat procesory, které podporují technologii NEON. Tato technologie umožňuje několikanásobně urychlit výpočet v mnoha oblastech, jako při zpracování obrazu, signálů nebo videa. Toto urychlení spočívá v instrukcích typu Single Instruction Multiple Data (SIMD). Tedy umožňuje zpracovávat větší množství dat paralelně. Informace v následujících kapitolách o NEON technologii vycházejí z [1].

3.2.1 Struktura NEON technologie

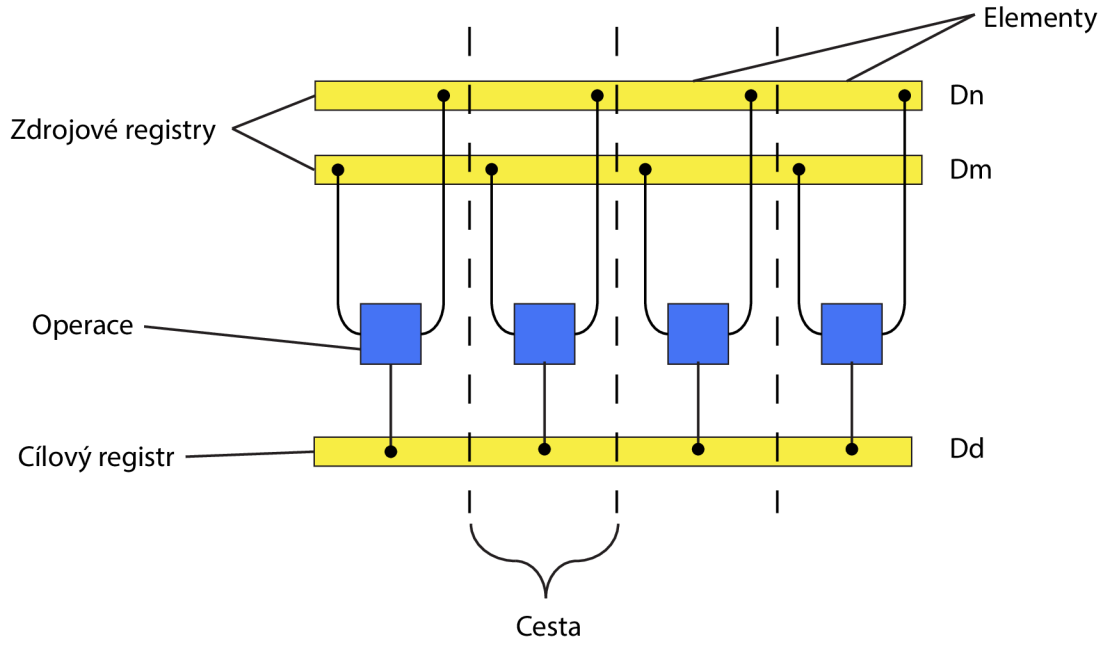
NEON technologie je 128 bitová SIMD architektura rozšiřující sérii procesorů ARM Cortex-A. Obsahuje nezávislou frontu instrukcí. Základem této technologie je nová sada registrů používaná v jednotce NEON. Registry se značí d0-d31 a mají 64 bitovou velikost. Tyto registry lze sloučit do registrů q0-q15, které jsou 128 bitové. Registry jsou chápany jako vektory prvků stejného datového typu. Tyto datové typy mohou být [1]:

- Znamenkové/Bezznaménkové
- 8,16,32,64 bitové
- Celočíselné/Desetinné

Pomocí NEON technologie je možné pracovat s pamětí, která je nějakým způsobem rozdělena, uceleně. Například pokud máme obrázek v RGB, a tedy v paměti jdou jednotlivé barevné složky za sebou, je možné pomocí interleavingu spojit jednotlivé barevné složky k sobě a pracovat s nimi jednotně [2].

3.2.2 NEON intrinsics

Pro zapsání kódu pro jednotku NEON je nutné zapsat kód ve specializovaných instrukcích, které procesoru řeknou, že tyto instrukce a data zpracuje jednotka NEON. Při psaní kódů ve vyšších programovacích jazycích je to velmi komplikované. Proto byl vytvořen NEON intrinsics. NEON intrinsics umožňuje psát NEON instrukce ve vyšších programovacích jazycích jako je C, C++ a jiné. O kompilaci do specializovaných instrukcí se postará kompilátor za nás. Pro aplikaci těchto funkcí musí být připojen soubor *arm_neon.h*. Základními funkcemi jsou Load a Store, které zajišťují naplnění NEON registrů a uložení dat z NEON registrů do paměti. Avšak existuje mnoho dalších funkcí, jako například pro matematické operace, posuvné operace, logické operace a mnoho dalších [1].



Obrázek 3.1: Struktura NEON jednotky. Inspirováno z [1].

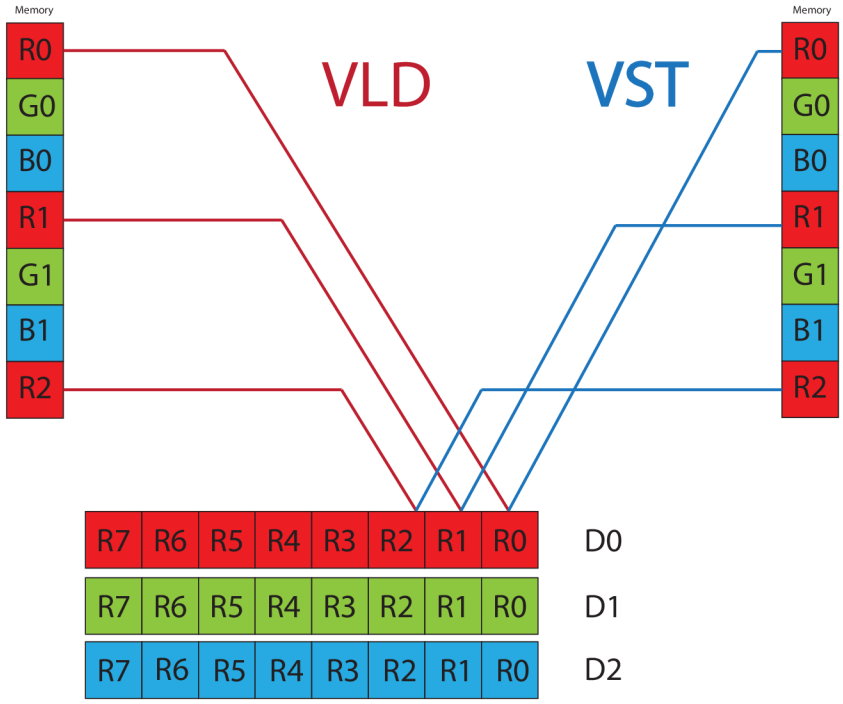
3.3 Nativní kód ve Windows Phone 8

Windows Phone 8 přinesl vývoj v nativním kódu. Tento vývoj v předchozí verzi nebyl podporován. Změna umožňuje výrazné zrychlení v určitých druzích aplikací, jako je zpracování obrazu, signálů či videa. Velkým přínosem je také možnost využít plný potenciál NEON technologie, kterou ve Windows Phone 7 nebylo možné použít, i když mobilní zařízení mělo procesor s NEON jednotkou. A v neposlední řadě je tato změna výhodná pro meziplatformní programování. Jelikož algoritmus v nativním kódu lze přenést na jiné platformy skoro beze změn. Nyní lze v nativním kódu (C++) použít NEON intrinsics a s jeho pomocí psát výrazně rychlejší algoritmy.

Windows Phone 8 plně podporuje jazyk C++ [20]. Tudíž je možnost znovu využít knihovny, které máme na Windows i do telefonu. Windows Phone 8 může také používat Windows Phone Runtime, což je knihovna poskytující různá API, jako například senzory, rozpoznávání a syntézu řeči, kontakty, baterie, kameru a další. Knihovna také zajišťuje práci s objekty, eventy, vyjímkami a dalšími částmi standartního programovacího modelu.

3.4 DirectX 11 ve Windows Phone 8

Windows Phone 8 podporuje vykreslování pomocí DirectX 11 [21]. S DirectX ve Windows Phone 8 se pracuje v nativním kódu a výsledky jsou zobrazovány pomocí *DrawingSurface* nebo *DrawingSurfaceBackgroundGrid*, což jsou XAML prvky. Rozdíl mezi těmito prvky je takový, že *DrawingSurfaceBackgroundGrid* je pod všemi ostatními XAML prvky, je tedy na pozadí. Oproti tomu u *DrawingSurface* záleží jen na nás, kam ho umístíme a kde se bude výsledek DirectX vykreslovat.



Obrázek 3.2: Ukázka interlevingu. VLD načítá data do jednotlivých registrů. VST ukládá data zpět do paměti tak jak byla. Inspirováno z [2].

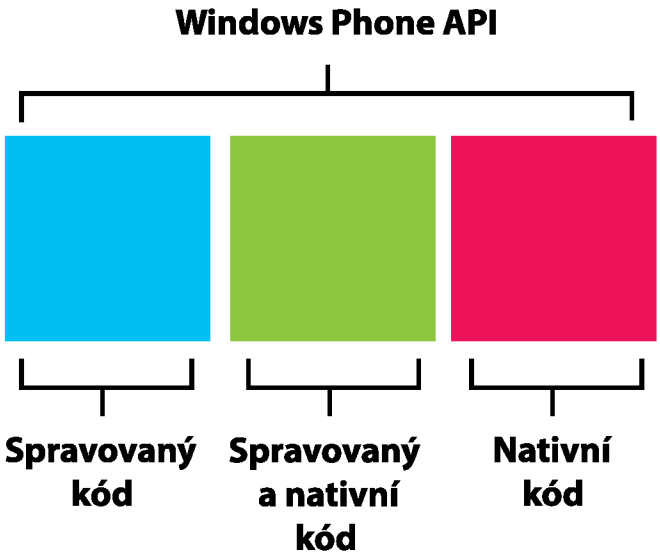
Následující informace vychází z knih DirectX - začínáme programovat [23], Beginning DirectX 11 Game Programming [25] a Managed DirectX 9: graphics and game programming: kick start [19].

3.4.1 Objekty a vertexy

V DirectX se jednotlivé objekty skládají ze 3 základních stavebních prvků:

- Vertexy
- Hrany
- Plochy

Z těchto prvků lze poté vytvořit libovolný tvar objektu. Vertex je obyčejný bod v prostoru. Souřadnice bodu jsou vztaženy většinou k nějakému centrálnímu bodu objektu. U krychle by to mohl být střed krychle a tedy jednotlivé souřadnice vertexů (vrcholů krychle) by nebyly souřadnicemi v obraze, ale pouze souřadnice vzhledem ke středu krychle. Kromě souřadnic může vertex obsahovat informace o barvě, textuře apod. Spojením dvou vertexů vznikne hrana a spojením hran dostáváme plochu. Plochy se obecně nazývají polygony, které nejčastěji bývají trojúhelníkového či čtvercového tvaru. Nejsnadnější práce je však s trojúhelníky, jelikož jsou vždy rovinné.



Obrázek 3.3: Jednotlivé části Windows Phone API. Inspirováno z [20].

3.4.2 Souřadnicový systém

DirectX používá trojrozměrný souřadnicový systém, kde jsou objekty reprezentovány svojí šířkou, výškou a hloubkou. Je zde využit kartézský pravoúhlý souřadnicový systém, kde poloha bodů je dána souřadnicemi na všech 3 osách (x,y,z). Směry jednotlivých os lze určit dle pravidla levé či pravé ruky. Pravidlo levé ruky říká, že když nasměrujeme levou ruku v kladném směru osy x a ohneme prsty, kromě palce, v kladném směru osy y, tak palec nám potom ukazuje kladný směr osy z. Totéž platí i pro pravidlo pravé ruky. DirectX obvykle využívá levoruký souřadnicový systém, avšak po příslušných úpravách lze používat v DirectX i pravoruký souřadnicový systém.

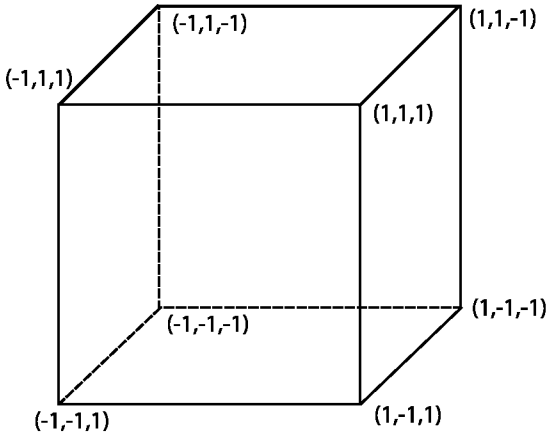
3.4.3 Transormace a matice

DirectX je nízkoúrovňový přístup ke grafické kartě a proto zde neexistuje pojem kamera jako v modelovacích softwarech. Pro práci s objekty, kamerou a obecně s celou scénou slouží transormace, což jsou matematické operace prováděné nad vertexy. Avšak ve scéně může být velké množství vertexů. Proto se využívá transformačních matic pro urychlení výpočtů. Za pomoci těchto transformací jsme schopni vytvořit 3D scénu a tuto 3D scénu zobrazit na 2D displeji.[23].

Existují tedy 2 druhy transformací. První jsou základní transformace a druhými jsou transformace pro zobrazení objektů na displeji. Základní transformace jsou:

- Posun
  - **Transformační matice posunu**

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ X & Y & Z & 1 \end{bmatrix}$$



Obrázek 3.4: Vrcholy krychle definované okolo středového bodu krychle.

- Změna měřítka
  - **Transformační matice změny měřítka**

$$\begin{bmatrix} X & 0 & 0 & 0 \\ 0 & Y & 0 & 0 \\ 0 & 0 & Z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotace
  - **Transformační matice rotace kolem osy x**

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos a & \sin a & 0 \\ 0 & -\sin a & \cos a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

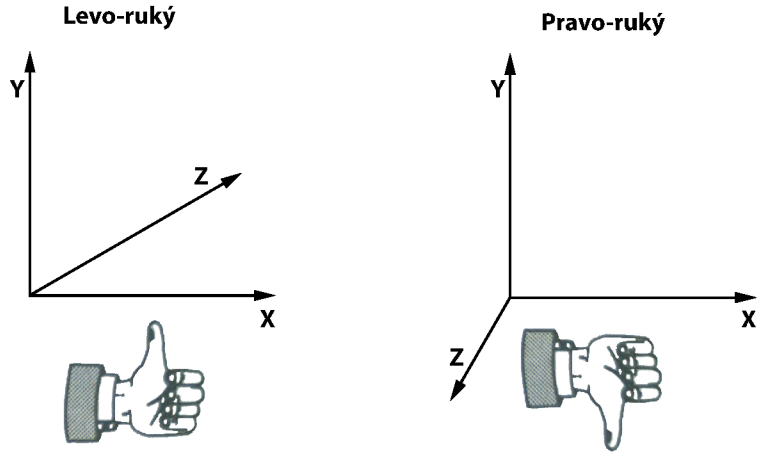
  - **Transformační matice rotace kolem osy y**

$$\begin{bmatrix} 1 & \cos a & -\sin a & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \sin a & cosa & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - **Transformační matice rotace kolem osy z**

$$\begin{bmatrix} \cos a & \sin a & 0 & 0 \\ -\sin a & \cos a & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pomocí základních transformací lze posouvat, rotovat či měnit velikost jednotlivých objektů.



Obrázek 3.5: Pravoruký a levoruký souřadnicový systém. Inspirováno z [19].

Transformace pro zobrazení se využívají pro převod jednotlivých prostorů do prostoru pro zobrazení na displeji. Těmito prostory jsou:

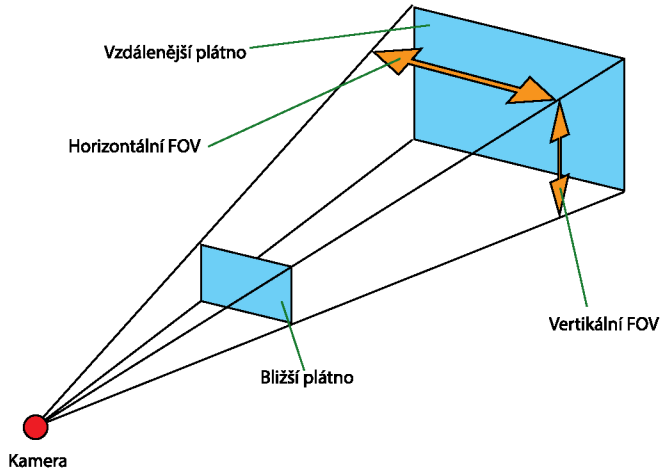
- Lokální prostor
- Světový prostor
- Pohledový prostor
- Projekční prostor
- Prostor displeje

Prvním prostorem je lokální prostor, což je prostor relativní k objektu. Když chceme vytvořit objekt, obvykle ho vytváříme okolo centrálního bodu (0,0,0), díky čemuž lze objekty vytvářet rychleji a efektivněji. To proto, že se nemusíme starat o pozice mezi jednotlivými objekty, ale pouze definujeme jeden objekt. Pokud bychom chtěli vytvořit les, bylo by nesmyslné vytvářet tisíce stromů. Takto vytvoříme jeden a jeho rozkopírováním dosáhneme kýženého efektu.

Další prostor je světový prostor. V tomto prostoru se umísťují jednotlivé objekty relativně vůči sobě, okolo centrálího bodu světa (0,0,0). K tomu slouží světové transformační matice, které jsou vytvářeny násobením základních transformačních matic (posunu, změny měřítka, rotace). Aplikací těchto transformačních matic na vertexy objektů dochází k transformaci lokálního prostoru na světový prostor. Po této transformaci jsou všechny objekty umístěny ve scéně správně vůči sobě.

Prostor pohledu je zjednodušeně řečeno kamerový prostor. V DirectX je kamera umístěna v bodě (0,0,0) a dívá se směrem dolů do osy z. Transformací světového prostoru do prostoru kamery dosahujeme efektu pohybu kamery ve scéně. Opak je pravdou, a tedy transformací se pohybuje svět před kameru, která je stále ve fixním bodě. Transformační matici pro transformaci světového prostoru do prostoru kamery lze vytvořit pomocí vestavěné funkce DirectX *XMMatrixLookAtLH* pro levoruký souřadnicový systém, *XMMatrixLookAtRH* pro pravoruký souřadnicový systém nebo pomocí základních transformací posunu, změny měřítka a rotace. Při použití posledně zmíněného je však nutností výslednou matici invertovat, jelikož se svět pohybuje okolo kamery a ne kamera okolo světa.

Prostor projekce je trošku jiný prostor než předchozí prostory. Tímto prostorem definujeme vše, co se má vykreslit a co naopak zahodit. Tento prostor je definován 6 stěnami a tvoří pyramidu s useknutým vrcholem. V místě vrcholu je umístěna kamera a pyramida poté tvoří prostor, ve kterém jsou objekty vykreslovány. V momentě, kdy je objekt mimo tento prostor, je zahozen, případně oříznut. Transformační matice je vytvářena pomocí vestavěné funkce *PerspectiveFovLH* či *PerspectiveFovRH*. Posledním prostorem je prostor



Obrázek 3.6: Prostor projekce. Inspirováno z [19].

displeje, tedy x a y souřadnice na displeji, kde 0,0 je levý horní roh displeje. Tento prostor není potřeba definovat, jelikož je to spíše myšlenka než fyzický prostor. Prostor displeje se využívá při zjišťování, zdali jsme dotekem na displej klikli na 3D objekt ve scéně.

3.5 Vývoj pro Windows Phone 8

Vývoj pro Windows Phone 8 má jistá úskalí. Abychom mohli vyvíjet pro tuto mobilní platformu, je nutné mít nainstalován operační systém Windows 8 a to v 64 bitové verzi. Velikost paměti RAM musí být minimálně 4 GB a musíme mít nejméně 6.5 GB volného místa na pevném disku. Emulátor Windows Phone 8 operuje jako Hyper-V virtuální stroj nad Windows 8. Z toho důvodu je nutná hardwarová podpora Hyper-V a 4 GB RAM [12].

3.6 Publikace na Store

Majitelé mobilního zařízení s Windows Phone 8 nemají žádnou možnost nahrávat si do mobilního zařízení vlastní aplikace. Každá aplikace musí být stáhnuta ze Microsoft Store. Tento způsob zajišťuje větší bezpečnost pro uživatele a také větší pohodlí při hledání nových aplikací.

Aby mohla být aplikace nahrána na Microsoft Store, musí nejprve projít certifikačním procesem. K tomu, aby aplikace tímto procesem korektně prošla, musí splňovat několik požadavků [12]. Po úspěšné certifikaci je aplikace podepsána certifikátem od Microsoftu a je publikována na Microsoft Store.

Kapitola 4

Návrh a implmentace knihovny pro detekci UMF a Demo aplikace

Cílem této práce je vytvořit knihovnu, jenž bude plnit funkci detektoru UMF a demo aplikace pro Windows Phone 8, která bude tuto knihovnu využívat. Demo aplikace by měla za pomoci této knihovny zobrazovat jednotlivé kroky detekce a také zobrazovat ladící informace. Případně může vykreslovat 3D objekt na markeru. Vlastní detekci UMF by měla zprosředkovávat knihovna, která je napsána v nativním kódu (C++), jelikož její hlavní úkol je zpracování obrazu, což je v nativním kódu mnohem rychlejší.

4.1 Architektura

Architektura celého projektu je zobrazena na obrázku 4.1. Jednotlivé vrstvy a rozhraní mezi nimi bude popsáno později.

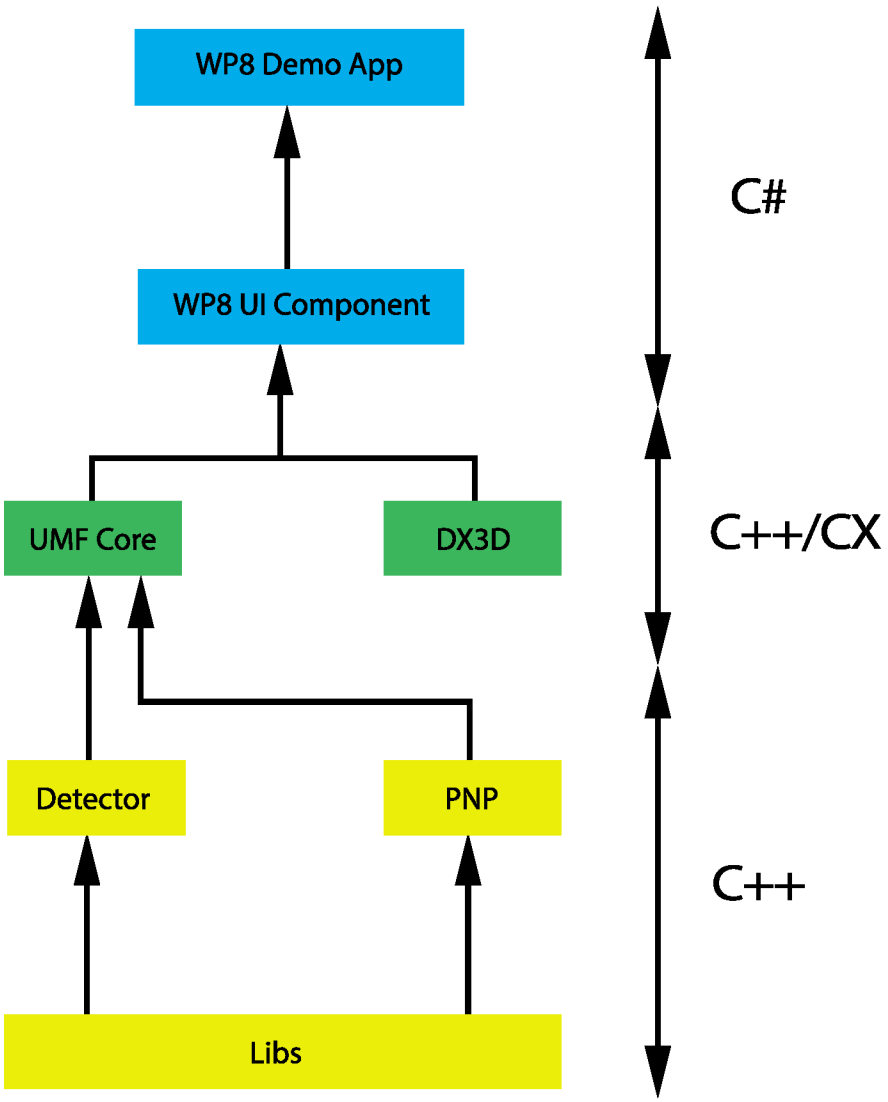
4.1.1 Libs

Začneme od nejnižší vrstvy. Tato vrstva poskytuje potřebné knihovny, které jsou využívány ve vrstvách vyšších, a tím je zajištěno, že se knihovna zbytečně neduplikuje. Je zde obsažena knihovna Eigen[6], která poskytuje lineární algebru, matice, vektory a spoustu dalších algoritmů. Jelikož detekce UMF hodně využívá pojmy jako matice, vektory, body apod., je tato knihovna velice užitečná. Díky ní je kód přehlednější, protože umožňuje různé operace nad všemi datovými typy jako např. vektory, kde lze použít pouze operátor sčítání pro sečtení dvou vektorů. Používá se zde jazyk C++, který se poté kompiluje jako statická knihovna *.lib*.

Knihovna Eigen umožňuje vektorizaci a zrychlení algoritmů. Konkrétně jsou podporovány následující SIMD instrukční sady:

- SSE, minimálně je vyžadováno SSE2
- AltiVec
- ARM NEON

Pro povolení vektorizace je nutné říci překladači, aby použil odpovídající instrukční sadu. Pro ARM NEON jsou to přepínače *-mfpu=neon* a *-mfloat-abi=softfp*. Bohužel se nepodařilo tyto přepínače nastavit v překladači pro Windows Phone 8. Knihovna Eigen sice ve WP8



Obrázek 4.1: Architektura aplikace.

funguje bez problému, ale úplná kompatibilita není oficiálně podporována a tedy vektorizace knihovny Eigen pomocí NEON jednotky není možná.

4.1.2 Detektor

Další vrstvou v celé architektuře je vrstva detektoru a opět je zde využit jazyk C++.Tento jazyk byl zvolen z toho důvodu, jelikož detekce je výpočetně náročná operace a v C++ by tato operace měla být rychlejší a efektivnější. V této části je prováděna vlastní detekce UMF a poskytovány výsledky detekce. A to jak výsledky detekce UMF, tak i ladící informace o detekci jako časy jednotlivých částí detekce atd. Vrstva detektoru využívá nižší vrstvu popsánou dříve, protože zde pracujeme hodně s vektory a maticemi, které obsahuje knihovna Eigen ve vrstvě Libs.

Detektor obsahuje metodu *bool UMFDetectorCore::DetectGrid*, která je vstupním bodem pro detekci. Návratovou hodnotou je *boolean* hodnota, která určuje úspěch či neúspěch

celého procesu. Vstupními parametry jsou:

- *byte* data*, což je vstupní snímek, na kterém se provádí detekce
- *byte* image*, což je výstupní snímek, který se poté zobrazuje na displeji a jsou do něj vykreslovány jednotlivé části detekce

Výstupními parametry jsou:

- *DetectedWindow* window*, což je výsledek detekce, obsahuje barvy, souřadnice detekovaného okna a pozici a orientaci okna v markeru
- *MeasureData* measuredata*, což jsou ladící informace obsahující časy jednotlivých částí detekce, počty nalezených edgelů a mnoho dalších informací

Jednotlivé části detekce a jejich implementace budou popsány později.

4.1.3 PnP

Perspective-n-Point Camera Pose Estimation je problém, který řeší určení pozice kamery v prostoru na základě n korespondencí bodů ze 3D do 2D. Tato vrstva poskytuje vstupní metodu *PnPSolver::computeCameraPose*, která provádí právě tuto operaci. Vstupními parametry jsou:

- *CorrespondenceSet correspondencesOrig*, což jsou korespondence bodů ze 3D do 2D
- *Eigen::Vector2i imageSize*, což je šířka a výška snímku
- *int flags*, což jsou parametry, kterými lze nastavit proces výpočtu
  - *PNP_FLAG_COMPUTE_CAMERA* - bude dopočítána rotace a pozice kamery
  - *PNP_FLAG_GL_PROJECTION_MV* - používá se pro OpenGL a proto v této práci není použit
  - *PNP_FLAG_SWAP_Y* - řeší problém inverzních hodnot Y, tedy že v obraze je bod (0,0) levý horní, kdežto pro DirectX levý spodní
  - *PNP_FLAG_RIGHT_HANDED* - úprava hodnot pro pravoruký souřadnicový systém
  - *PNP_FLAG_USE_LAST* - použít poslední výpočet pro následující iteraci

Výsledkem celého algoritmu je *Quaternion*, což je rotace a pozice. Princip tohoto postupu je takový, že zjistíme význačné body ve sledovaném obraze (snímku) a k těmto bodům známe odpovídající hodnoty v modelu. V našem případě jsou význačnými body v obraze středy čtverců detekovaného okna a body modelu jsou známy z modelu UMF markeru. Ná základě těchto hodnot je vytvořena odpovídající korespondence, s jejíž pomocí je dopočítána pozice a rotace kamery v prostoru. Tato knihovna vychází z EPnP citeEPNP a využívá knihovnu OpenCV [4]. Bohužel OpenCV není kompatibilní s Windows Phone 8.Proto bylo třeba OpenCV zaměnit za knihovnu Eigen [6], která poskytuje obdobné matematické operace jako OpenCV, které jsou pro výpočet pozice a rotace kamery nutné. Tato záměna byla provedena panem Istvánem Szendrazsim [7] a zároveň jím byl přidán další způsob výpočtu, který lze použít samostatně nebo i v kombinaci s již zmíněným EPnP.

4.1.4 UMF Core

Následující vrstva již není napsána v čistém jazyce C++, ale v rozšíření jazyka C++ o prvky Windows Phone 8. Tento jazyk je označován jako C++/CX (Component eXtensions). Konkrétně se jedná o rozšíření, která umožňují psát aplikace a komponenty pro Windows Runtime nebo Windows Store.

Syntaxe odpovídá C++/CLI, avšak cílová platforma je Windows Runtime. Díky tomu je možné intereagovat se spravovaným kódem jako je C#, Visual Basic, JavaScript a dalšími jazyky, které jsou podporované ve Windows Runtime.[17]

Vrstva odděluje čisté C++, tedy nespravovaný kód od spravovaného kódu. Vytváří určité rozhraní mezi jazyky C++ a C#. Díky tomuto oddělení tato vrstva poskytuje dvě události. První je událost úspěšné detekce a druhá úspěšného určení pozice a rotace kamery v prostoru.Těmito událostmi jsou informovány další vrstvy o úspěšné detekci a mohou s předanými informacemi dále nakládat dle svého uvážení.

Aby bylo možné pracovat v této vrstvě s vrstvami nižšími, které jsou v čistém C++, bylo nutné vytvořit třídu, která s těmito vrstvami pracuje a této třídě bylo třeba při překladu nastavit, aby nebyly přilinkovány knihovny Windows Runtime. Pokud by překladač nebyl takto informován, docházelo by k chybám při překladu. Tato třída *Core* obsahuje dvě metody, kde každá pracuje s jednou nižší vrstvou. První z nich je metoda *Detect*, která pracuje s vrstvou detektoru a druhou je metoda *PnP*, která pracuje s vrstvou PnP. Třída poskytuje také další metody pro nastavení detektoru, načtení markeru atd.

Pro korektní funkčnost této vrstvy je nutné správně nastavit několik vlastností, které jsou zde obsaženy.

- *PhotoCaptureDeviceĈaptureDevice* - jedná se o vlastnost, do které se nastavuje třída s otevřenou kamerou, jelikož se pak kamera používá pro získávání snímku k detekci
- *OutputBuffer* - jedná se výstupní buffer, ze kterého se berou výsledné snímky a zobrazují se na displeji
- *OutputBufferSize* - jedná se o velikost výstupního bufferu

Jakmile jsou výše uvedené vlastnosti korektně nastavené, je možno používat metodu *IAsyncActionĜetNewFrameAndApplyEffect*. Výstupním parametrem je třída *IAsyncAction^*. Toto rozhranní zajišťuje asynchronní operace. Respektive UI vlákno nemusí neustále čekat na dokončení této metody, ale je informováno po jejím dokončení. Díky tomu jsou snímky zpracovávány paralerně. Cílem této metody je získat snímek z kamery, převést ho do černobílé podoby a předat nižší vrstvě, která se postará o detekci. Po úspěšné detekci je vyvolána příslušná událost, ve které jsou předány informace o detekci a ladící informace.

4.1.5 DX3D

Tato vrstva má na starost vykreslování 3D objektů do scény. Sice je také napsána v C++/CX, tedy v jazykovém rozšíření jazyka C++, ale nevytváří žádnou mezivrstvu. Jsou zde obsaženy metody, kterými lze nastavovat vykreslování. Toto nastavení provádí vrstva *WP8 UI Component*, která po úspěšném určení pozice a rotace kamery předá rotační quaternion a pozici právě do této vrstvy.

Základem je zde třída *Direct3DInterop*, která je vstupním bodem do této vrstvy. Obsahuje metodu *CreateContentProvider*, která vrací objekt, implementující rozhraní *IDrawingSurfaceContentProvider*. Toto rozhraní je vyžadováno při nastavení poskytovatele obsahu (Content provider) v UI prvku *DrawingSurface*.

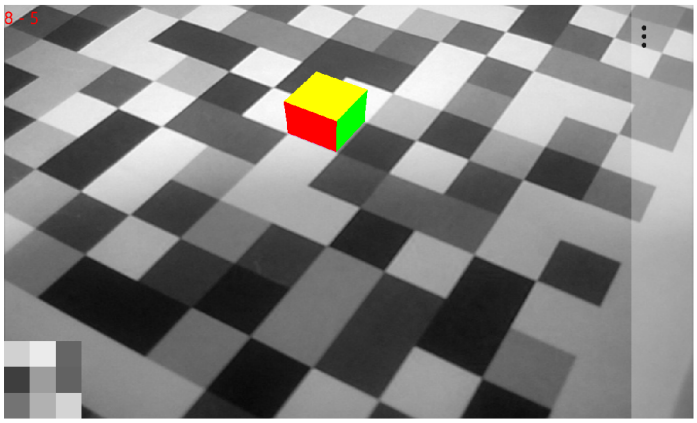
Dále je zde třída *CubeRenderer*, ve které je definován 3D objekt, v tomto případě krychle. Tato krychle je definována pomocí vertexů, ze kterých je poté vytvořena 3D krychle. Při každé aktualizaci scény se vytváří světová a pohledová transformační matice. Zatímco světová transformační matice je maticí jednotkovou, jelikož objekt necháváme tam, kde je, tak pohledová matice se vytváří z vypočítané pozice a rotace kamery. Pro rotaci je použit převod quaternionu na rotační matici a to pomocí vestavěné funkce *XMMatrixRotationQuaternion*. Obdobně je na tom pozice, která se převádí na translační matici pomocí *XMMatrixTranslation*. Výsledkem jsou dvě matice. Rotační matice R a translační matice T. Výsledná pohledová transformační matice V se dopočítá dle následující rovnice [19]:

$$V = (R \cdot T)^{-1} \tag{4.1}$$

Výsledek násobení se musí invertovat proto, že v DirectX je kamera stále na místě a svět se pohybuje okolo kamery. Proto, když chceme pohnout kamerou, musíme pohybovat světem pomocí invertované matice.

4.1.6 UI Component

Tato vrstva je již napsána zcela ve spravovaném kódu, a to konkrétně v jazyce C#. Obsahem vrstvy je komponenta uživatelského rozhraní, která zastřešuje všechny vrstvy pod sebou a tím zprostředkovává uživateli jednoduchou možnost využití ve své vlastní aplikaci. Tato komponenta je klasická XAML komponenta, kterou lze jednoduše vložit do libovolné XAML stránky aplikace pro Windows Phone 8. Komponenta obsahuje prvek *MediaElement*, který zobrazuje video z kamery s vykreslenými dodatečnými informacemi z průběhu detekce, již zmíněný *DrawingSurface*, ve kterém se zobrazuje 3D objekt vykreslovaný pomocí DirectX 11 ve DX3D vrstvě, jak je vidět na obrázku 4.2. Pozice a orientace detekovaného okna v markeru se v této komponentě zobrazuje jednak v textové podobě, tak i graficky, kde v levém dolním rohu se vykresluje detekované okno markeru. Proč byl vybrán pro zobrazení videa z kamery právě prvek *MediaElement* a jeho princip, je popsáno v 4.2



Obrázek 4.2: Vykreslený 3D objekt.

V komponentě jsou zaregistrovány obě události, které poskytuje nižší vrstva *UMF Core*. Ve chvílí, kdy je vyvolána událost úspěšné detekce okna v markeru, získají se informace o výsledku detekce. Poté se nastaví UI prvky na detekovanou pozici a orientaci okna v markeru a taktéž se ze získaných barev čtverců detekovaného okna nastaví jednotlivé UI prvky, které detekované okno reprezentují v komponentě. Ve chvíli, kdy je vyvolána druhá událost

úspěšného určení pozice kamery v prostoru, je získán rotační quaternion a pozice kamery a těmito hodnotami jsou aktualizovány parametry v *DX3D* vrstvě, která se stará o vykreslení 3D objektu na displeji.

4.1.7 Demo

V poslední a nejvyšší vrstvě celé architektury je vlastní Windows Phone 8 aplikace, která využívá výše zmíněné komponenty. Hlavní funkcionalita demo aplikace je následovná:

- Vykreslit 3D objekt - toto je zajištěno za pomoci již zmíněného DirectX a výpočtu pozice a orientace kamery v prostoru.
- Zobrazit jednotlivé fáze algoritmu - mezi jednotlivými částmi algoritmu lze přepínat za běhu. Pro přepnutí slouží tah prstem po displeji nahoru či dolů.
- Vypsání naměřených hodnot - jde o zobrazení časů detekce a další informacích, které byli zjištěny při detekci.

Vytvořená demo aplikace umožňuje zobrazení jednotlivých kroků detekce UMF:

- Zobrazení rozkladových řádků
- Vykreslení detekovaných edgelů jako bodů
- Vykreslení detekovaných edgelů jako bodů a rozkladových řádků
- Vykreslení detekovaných edgelů s jejich odhadnutým směrem dle Sobelova operátoru
- Vykreslení detekovaných edgelů s jejich odhadnutým směrem dle Sobelova operátoru a rozkladových řádků
- Vykreslení upřesněných přímek
- Vykreslení obou skupin přímek rozdělených pomocí metody RANSAC
- Zobrazení dvou hlavních úběžníků scény
- Vykreslení vějířů
- Vykreslení bodů, které určují středy čtverců
- Zobrazení 3D objektu

Demo aplikace pouze nastavuje nižší vrstvy, aby tyto informace vykreslovala. V demo aplikaci je možné vybrat si, zdali detekce bude probíhat pomocí kamery realtime, nebo pomoci načteného obrázku. Dále demo aplikace zobrazuje časy jednotlivých kroků detekce, které jí poskytne nižší vrstva. A v neposlední řadě také zobrazuje umístění detekovaného okna v UMF.

Funkce demo aplikace jsou rozděleny do několika obrazovek. Na první je obraz z kamery, ve kterém je možné zobrazit jednotlivé kroky detekce. Zobrazení detekovaného okna v levém dolním rohu obrazovky a pozice a orientace v markeru. Mezi jednotlivými částmi algoritmu lze přepínat pouhým přejetím prstu po displeji ve směru nahor či dolů. Na další obrazovce se nacházejí jednotlivé časy detekce a další informace získáné při detekci. Tyto informace o detekci jsou získávány pomocí události *Detector_OnDetected*, která je vyvolávána

Obrázek 4.3: Panorama Control ve Windows Phone 8 [15].

UI komponentou a zobrazí je. A na poslední obrazovce je nastavení parametrů detekece. Prvním z parametrů je offset rozkladových řádků v pixelech a druhým je rozdíl barev pro porovnování.

Uživatelské rozhraní Windows Phone 8 je založeno na *Pivot Control* a *Panorama Control*. Pro zobrazení výše zmíněných obrazovek jsme vybrali *Pivot Control*, jelikož Panorama Control vykresluje všechny obrazovky najednou a to zpomaluje běh aplikace,*Pivot Control* vykresluje jen aktuální *Pivot*, přestože poskytuje velmi podobnou funkcionalitu jako *Panorama Control* [13].

4.2 Zpracování snímku z kamery

Pro zobrazení snímků z kamery na displeji lze ve Windows Phone 8 použít *VideoBrush*. Tento prvek se definuje v UI aplikaci, tedy v XAML.

```
<Grid x:Name="LayoutRoot" Background="Transparent">
    <Rectangle Width="640" Height="480" Canvas.ZIndex="1">
        <Rectangle.Fill>
            <VideoBrush x:Name="viewfinderBrush" />
        </Rectangle.Fill>
    </Rectangle>
</Grid>
```

Poté se mu nastaví zdroj dat, což je kamera.

```
Windows.Foundation.Size resolution = new Windows.Foundation.Size(640, 480);
m_camera = await PhotoCaptureDevice.OpenAsync(CameraSensorLocation.Back, resolution);
ViewfinderBrush.SetSource(m_camera);
```

Tento postup je velmi jednoduchý, avšak není možné po získání snímku z kamery, tento snímek upravit a až poté ho zobrazit na displeji, což je pro tuto práci klíčové. Pro tuto funkcionalitu je třeba využít jiné techniky.

Jednou z nich je využití Image control a WriteableBitmap jako zdroje. Pak vypadá zapis následovně [22]:

```
<Image x:Name="MyCameraViewfiner"  Width="640" Height="480">
```

```
Deployment.Current.Dispatcher.BeginInvoke(delegate()  // Switch to UI thread
{
    ARGBPx.CopyTo(wb.Pixels, 0);
    wb.Invalidate();
});
```

V tomto případě sice již jsme schopni přistupovat ke snímku dřív, než se zobrazí na displeji, ale vznikne jiný problém. Jak je vidět v příkladě výše, pro aktualizaci snímku na displeji je nutné se přepnout na UI vlákno. Přepínání vláken je obecně velmi pomalé a navíc se snímek neaktualizuje, dokud se s ním bude na UI vlákně pracovat. Proto tento způsob také není vhodný [22].

Další možností je použít DirectX texturu. To je sice nejrychlejší způsob, ale nabízí jen možnost vykreslovat. Pokud bychom chtěli získat nějaké informace jako barvy čtverců či jiné informace, není to možné.

Poslední a námi zvolená možnost je použití MediaElement. MediaElement je UI prvek, který lze použít pro přehrávání audia, videa a dalších. Tento prvek je schopen obstarat veškeré záležitosti s přehráváním a je velmi optimalizovaný [22].

```
<MediaElement x:Name="MyCameraMediaElement" IsHitTestVisible="False"
              Margin="4" Width="640" Height="480" />
```

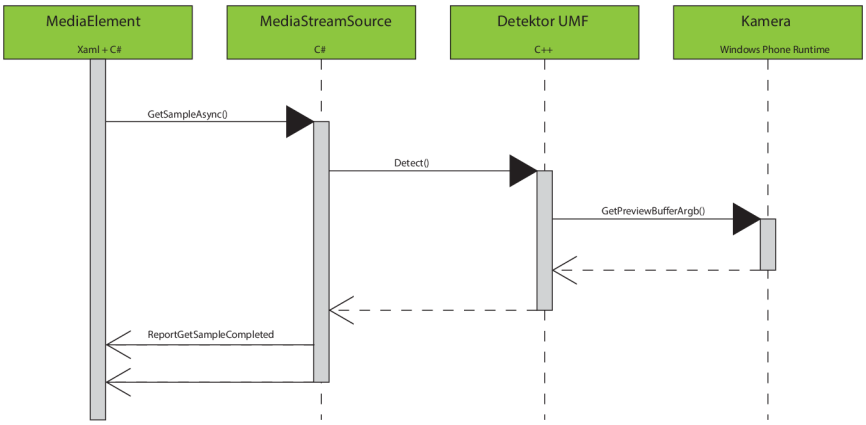
Abychom mohli tento prvek použít pro zpracování snímků z kamery, musíme definovat vlastní MediaStreamSource, kterým budeme plnit video snímky pro MediaElement. Třída MediaStreamSource definuje jeden stream, který je typu video (RGBA). Jelikož náš reálný zdroj je kamera, je třeba nastavit MediaStreamSource na nekonečnou délku a zakázat vyhledávání [22].

```
mediaStreamAttributes[MediaStreamAttributeKeys.VideoFourCC] = "RGBA";
mediaStreamAttributes[MediaStreamAttributeKeys.Width] = _frameWidth.ToString();
mediaStreamAttributes[MediaStreamAttributeKeys.Height] = _frameHeight.ToString();
mediaStreamDescriptions.Add(_videoStreamDescription);

mediaSourceAttributes[MediaSourceAttributesKeys.Duration] =
TimeSpan.FromSeconds(0).Ticks.ToString(CultureInfo.InvariantCulture);
mediaSourceAttributes[MediaSourceAttributesKeys.CanSeek] = false.ToString();
```

MediaElement volá metodu GetSampleAsync() z MediaStreamSource vždy, když potřebuje další video snímek. Tato metoda je asynchronní, proto MediaElement nemusí čekat na její dokončení a díky tomu se UI vlákno nepozastaví. Po dokončení zpracování každého snímku je volána metoda ReportGetSampleCompleted, kterou se MediaElement dozví, že byl snímek zpracován a může ho zobrazit. Jednotlivé snímky jsou zpracovávány ve vlastních vláknech, což urychluje celé zpracování. Na obrázku 4.4 je sekvenční diagram, který zobrazuje princip zpracování snímku z kamery a zobrazení.

Vždy, když je zavolána metoda *GetSampleAsync()*, volá se metoda pro detekci, která je umístěna v nižší vrstvě architektury. Aby však mohla být detekce provedena a výsledný obraz předán k zobrazení prvku MediaElement, musí se provést několik kroků.

Nejdříve se musí nastavit vlastnost *CaptureDevice*, která obsahuje objekt s otevřenou kamerou. Za pomocí tohoto objektu se poté volá metoda pro získání snímku z kamery. Také se zde nastaví rozlišení pořizovaných obrázků. Dalším krokem je získání bufferu, do kterého se budou vykreslovat výsledné snímky pořízené kamerou. To se provede nastavením



Obrázek 4.4: Sekvenční diagram pro MediaElement. Inspirováno z [22].

vlastnosti *OutputBuffer*. V metodě *set* této vlastnosti je převední třídy *IBuffer* na *byte**, tedy pole bytů. A to z toho důvodu, jelikož v nižších vrstvách se pracuje v C++ a tedy není možno použít třídy C++/CX. Po těchto krocích již je možné využívat detekci markeru UMF.

Vlastní detekce se poté skládá z několika částí. První z nich je získání snímku z kamery. Tento snímek se poté převede do černobílé podoby. Pro tento převod byla využita technologie jednotky NEON, která několikanásobně tento krok zrychluje. Následně je volána metoda detektoru, které je předán černobílý snímek a objekt pro vykreslení jednotlivých částí detekce. Po úspěšné detekci je vyvolána událost *OnDetected*, ve které se předají výsledky detekce a hodnoty měření. Následujícím krokem je volání metody *_core.PnP*, které se předají detekované výsledky. Pokud metoda skončí úspěšně, získáme rotační quaternion a pozici kamery a vyvolá se událost *OnPoseEstimated*, ve které se předají získaná data.

4.3 Rozhraní detektoru

Knihovna s detektorem UMF poskytuje rozhraní, za jehož pomoci může libovolná aplikace s detektorem komunikovat. Součástí tohoto rozhraní je inicializace knihovny. Tedy prvotní nastavení vstupních parametrů detektoru. Knihovna poskytuje výsledné časy měření a především barvy středů detekovaného UMF.

Knihovna pro detekci markeru UMF je obsažena ve vrstvě *Detecor*. Jsou zde umístěny metody pro nastavení rozlišení snímků, metoda pro načtení markeru či změny stavu, ve kterém se detektor nachází. Na základě tohoto stavu se provadí pouze část detekce. Základním nastavením je provedení celého algoritmu, ale uživatel může změnou stavu zajistit provádění jen části algoritmu a tuto část poté nechat vykreslit. Například uživatel přepne detektor do stavu *VanishingPoints*, ve kterém se provádí pouze detekce úběžníků scény a tyto úběžníky se poté vykreslují. Vstupním bodem celého detektoru je metoda *DetectGrid*, která zajišťuje vlastní detekci. Vstupními parametry této metody je pole bytů, ve kterých je vstupní snímek z kamery, pole bytů , do kterých se vykreslují další informace. Výstupními parametry jsou dvě struktury. První předává výsledek detekce a druhá výsledky měření. Podrobný postup detekce a jednotlivých částí algoritmu bude popsán v další části této práce.

4.4 Detekce UMF

V kapitole 2.3 jsou vypsány jednotlivé části detekce UMF. Dále budou jednotlivé kroky detekce UMF popsány podrobněji.

4.4.1 Detekce edgelů

Jelikož snímky pořízené kamerou z mobilního zařízení jsou často velmi zašuměné a rozmazané, je třeba s tímto jevem při detekci počítat. Gradient hrany v takových snímcích je velmi široký a proto se při detekci hledá maximum (vrchol) gradientu pomocí adaptivního prahování.

Detekce edgelů je první krok v detekci UMF. Nejdříve se hledají hrany jednotlivých rozkladových řádků. Každá rozkladový řádek se prochází od začátku do konce a pomocí adaptivního prahování se dvěma okny s plovoucím průměrem se hledají hrany. Toto hledání je zobrazeno na obrázku 4.5. Okna jsou umístěna za sebou a do obou se vejdou 3 pixely. Hledaní hran probíhá tak, že do prvního okna se postupně vkládá aktuální pixel. Jakmile se první okno naplní, posledni pixel z tohoto okna se posune do druhého okna a z prvního se odstraní. Ve chvíli, kdy se naplní obě okna, dochází ke sledování absolutní hodnoty rozdílu oken. Jakmile tento rozdíl přesáhne práh $T = 25$, začne se hledat maximální rozdíl oken, dokud rozdíl neklesne pod práh.

V detektoru je tato část implementována ve třídě *FindEdgels*, jejíž vstupním bodem je metoda *FindEdges*. Jak již bylo uvedeno, nejdříve se prochází horizontální rozkladové řádky. Jelikož v poli pixelů jsou pixely uloženy řádek po řádku, je tato část snadnější. Stačí v každém kroku k aktuální pozici v poli přičíst hodnotu 1. Pro procházení vertikálních rozkladových řádků je třeba se posouvat po hodnotě 1 v ose y. Toho je docíleno tím, že při posunu o pixel není přičítána k aktuální pozici pouze hodnota 1, ale hodnota, která určuje počet pixelů v řádku. Tím docílíme, že se posuneme o jeden pixel v ose y. Dále je nutné, aby po dokončení procházení jednoho vertikálního rozkladového řádku byla pozice vynulována a nastavena na další rozkladový řádek. Kdybychom pouze přičetli další řádek, dostali bychom se mimo pole.

Metoda umožňuje nastavit offset rozkladových řádků (defaultně 50) a také hodnotu rozdílu barev (defaultně 25), dle kterého se určuje hrana. Výsledkem je seznam edgelů, konkrétně byl využit datový typ *vector*;

Obrázek 4.5: Hledání hran pomocí 2 oken s plovoucím průměrem [24].

.

Edgel je definován detekovanou hranou a směrem hrany. Směr hrany lze určit pomoci Sobelova operátoru. Tento operátor byl upraven tak, že pro odhad směru potřebuje pouze 4 body z obrazu. Původní operátor jich potřebuje 8, avšak pro naše účely postačuje upravený

Sobelův operátor. Sobelův operátor se skládá ze dvou částí. První je odhad h ve směru osy x a druhý v ve směru osy y.

$$h = (\ -1\ 0\ 1\ ) \quad v = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \tag{4.2}$$

Z výsledků Sobelova operátoru lze určit vektor gradientu $\vec{g_v}$. Tento vektor je kolmý na hranu a je tedy potřeba určit i normálový vektor $\vec{g_{vn}}$, který určuje směr hrany, tedy přímky procházející hranou markeru. Oba vektory jsou poté normalizovány, tak aby $|\vec{g_v}| = |\vec{g_{vn}}| = 1$

$$\vec{g_v} = (h, v) \quad \vec{g_{vn}} = (-v, h) \tag{4.3}$$

Použití Sobelova operátoru pro odhad směru hrany je velmi hrubé a nepřesné a je nutné odhadnutý směr hrany dále upřesnit.

4.4.2 Upřesnění odhadu směru přímek

Upřesnění odhadu směru přímky je prováděno hledáním celé přímky, které odpovídá nalezený edgel. Celá přímka se hledá postupným hledáním dalších hran ve směru odhadu. Máme bod p_0, což je nalezený edgel. Posunutím o w v odhadovaném směru s_n dostaneme nový bod X_i. Jelikož odhad Sobelovým operátorem je opravdu velmi nepřesný, je vhodné ze začátku hledat následující hranu blíže k nalezenému edgelu. Proto se w_i vypočítá po každé iteraci nově.

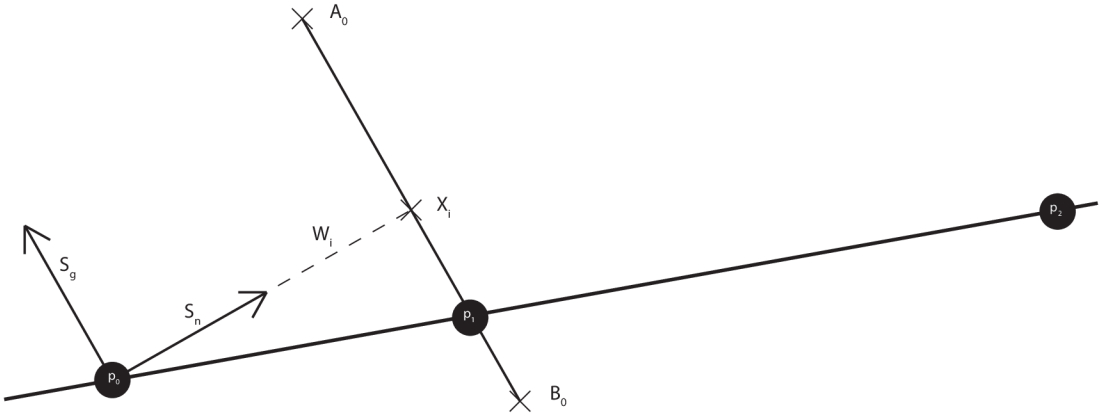
$$\begin{aligned} \tilde{X}_{i+1} &= p_i + w_i \hat{s}_i \\ w_{i+1} &= w_i + (10i + 5) \\ w_0 &= 0 \end{aligned} \tag{4.4}$$

Z tohoto bodu X_i vypočteme další dva nové body A_0 a B_0, kde vzdálenost mezi A_0 a B_0 je w_g. Poté pomoci půlení intervalu mezi těmito body hledáme další hranu p_{i+1}.

$$\begin{aligned} A_0 &= p_i + w_g \hat{s}_g \\ B_0 &= p_i - w_g \hat{s}_g \\ w_g &= \frac{w}{2} \end{aligned} \tag{4.5}$$

Toto upřesňování se opakuje několikrát po sobě a to v obou směrech (i záporná i). Přímka je prohlášena za vhodnou, pokud počet upřesnění je větší než 3. Upřesňování je ukončeno ve chvíli, kdy již mezi body A_0 a B_0 není žádná hrana, nebo směr nové hrany se příliš liší od původního směru.

Tato funkcionalita je implementována ve třídě *FindDirections*. Pro práci s body a vektory je využita knihovna Eigen [6], která umožňuje provádět aritmetické operace s body a vektory. Nebylo tedy nutné vytvářet vlastní metody. Aby nedocházelo k chybám, je zde implementováno několik kontrol. První je test, zda odhad Sobelova operátoru je číslo, tedy odhad není nekonečný, či nedošlo k jinému problému při odhadu za pomoci Sobelova operátoru. Pokud není, je výpočet ukončen a edgel je označen jako nepoužitelný. Délka přímky při upřesňování směru je omezena na délku maximalně 500 pixelů. Také je kontrolováno, zda další bod na přímce již není mimo obraz. Pokud ano, předčasně se ukončí výpočet. Toto ukončení však neznamená, že se daný edgel označí jako nepoužitelný. Je nutné zjistit, zda v průběhu došlo alespoň ke 3 úspěšným upřesněním. Edgel je označen jako nepoužitelný,



Obrázek 4.6: Upřesnění směru hrany. Inspirováno z [24].

pokud nedosáhl alespoň 3 upřesnění. Jestliže mezi vypočtenými body A_0 a B_0 není žádná hrana, tedy rozdíl barev je menší než 25, přeskočíme fázi půlení intervalu, tedy hledání další hrany a posuneme se dále a pokus opakujeme. Důvodem je to, že edgel může být detekován hned vedle dvou čtverečků, které mají shodnou barvu. A tedy při posunu na ně by nedošlo k nalezení hrany, přestože dále by hrana již nalezena byla. Proto se pokračuje v detekci, ale tento pokus o upřesnění není zaznamenán jako úspěšný a je nutné nalézt další úspěšné upřesnění. Jakmile je nalezena další hrana ve směru odhadu, je tato hrana testována, jestli se její směr příliš neliší od původního. To se provádí opět za pomoci Sobelova operátoru. Původní vektor $\vec{v_1}$ a nový vektor $\vec{v_2}$ jsou normalizovány. Potom je možné vypočítat úhel mezi těmito vektory a pokud je tento úhel větší jak 45°, tak potom je edgel označen jako nepoužitelný. Jelikož nový odhad může být otočen o 180°, tak je tento test proveden i s otočeným vektorem $\vec{v_2'}$.

$$\begin{aligned} \vec{v_1} &= (h, v) \\ \vec{v_2} &= (v_n, h_n) \\ \vec{v_2'} &= (-v_n, -h_n) \\ \cos^{-1}(\vec{v_1} \cdot \vec{v_2}) &> 45° \\ \cos^{-1}(\vec{v_1} \cdot \vec{v_2'}) &> 45° \end{aligned} \tag{4.6}$$

4.4.3 Rozdělení přímek do 2 hlavních skupin

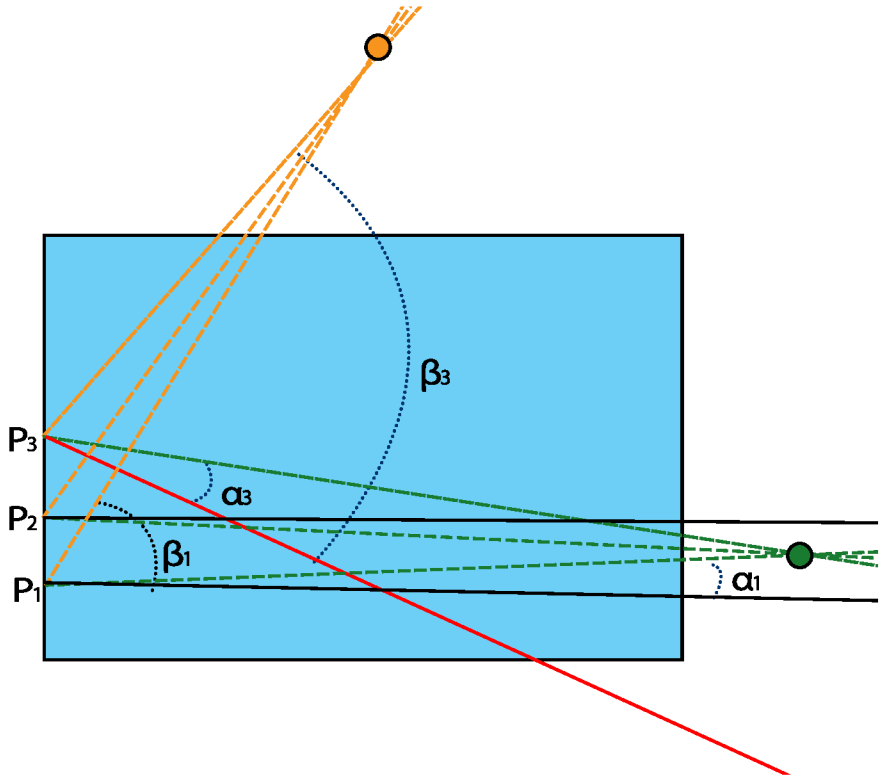
Pro nalezení dvou hlavních úběžníků je třeba rozdělit přímky do skupin, kde každá skupina odpovídá jednomu z hlavních úběžníků scény. Toto rozdělení je provedeno metodou RANSAC [26], což je iterační metoda sloužící k hledání prvků, které odpovídají definovanému modelu v zadané množině prvků. V zadané množině se obecně vyskytují jak vyhovující prvky (inliers), tak i prvky, které modelu neodpovídají, tzv. šumové prvky (outliers). Metoda RANSAC je nedeterministická a tedy k výsledku dospěje pouze s určitou pravděpodobností. Čím více iterací metoda provede, tím víc se zvyšuje pravděpodobnost dobrého výsledku. Princip algoritmu je následovný:

- Iterujeme přes odhadnutý počet iterací

- Náhodně vybereme prvky ze zadané množiny a odhadujeme model
- Pro prvky z množiny testujeme, zda patří do modelu, tedy jestli jsou inliers
- Uložíme počet outliers
- Po dokončení všech iterací je vybrán model s nejmenším počtem outliers

V této práci v metodě RANSAC náhodně najdeme dvě přímky z celé množiny upřesněných přímek. Potom procházíme všechny přímky a kontrolujeme, ke kterému směru vybraných přímek se směr té aktuální podobá. Přiřazení přímek ke správné skupině se provádí na základě skalárního součinu vektorů. Všechny vektory jsou opět normalizovány a tedy na základě skalárního součinu dvou vektorů je možné určit správnou skupinu. Pokud se nepodobá ani jedné, je zařazena do skupiny outliers. Kvalita modelu metody RANSAC je pak ohodnocena dle počtu outliers. Čím menší počet outliers, tím lepší model. Nakonec je vybrán model s nejmenším počtem outliers a z modelu dostaneme dvě skupiny přímek, kde každá odpovídá jednomu z hlavních úběžníků scény.

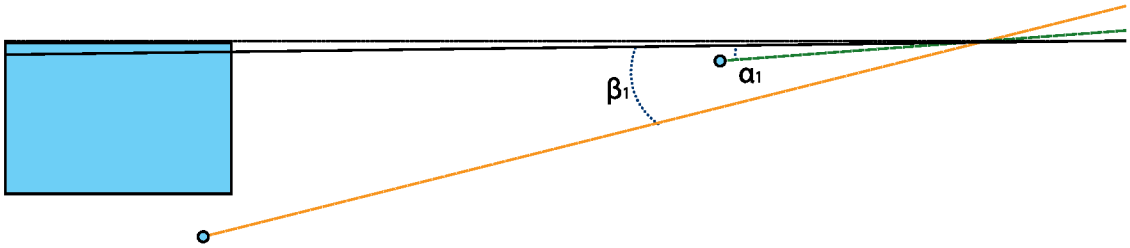
Z těchto skupin jsou vypočteny oba úběžníky. Výpočet není úplně přesný, jelikož se mohou dostat do těchto skupin i přímky, které znepřesňují tento výpočet. Proto je provedeno další roztřídění přímek do skupin, kde každá skupina odpovídá jednomu úběžníku scény.



Obrázek 4.7: Černé plné čáry jsou dobré přímky, kde jeden z úhlů je menší než 11°. Jak je vidět úhel α_1 je menší než 11° a zároveň je menší než β_1. Červená přímka je typickým příkladem přímky, která zkresluje výslednou pozici úběžníku a je třeba ji filtrovat. Úhel α_3 i β_3 jsou mnohem větší než 11° a proto je tato přímka zahozena.

Toto roztřídění však probíhá ve fázi, kdy známe přibližné polohy obou úběžníků a tohoto je zde využito. Z každého edgelu je vytvořena obecná rovnice přímky l_i a z této

rovnice je dopočítán průsečík P s osou Y. Poté za pomocí bodu P a obou úběžníků U_1 a U_2 jsou vytvořeny vektory $\vec{v_{1i}}$ a $\vec{v_{2i}}$, které definují přímky prochazející bodem P a daným úběžníkem. Poté se opět za pomocí skalárního součinu dopočítá úhel mezi vektorem přímky $\vec{v_i}$ a jednotlivými vektory z bodu P do úběžníků $\vec{v_{1i}}$ a $\vec{v_{2i}}$. Na základě výsledných úhlů α_1 a β_1 je daná přímka přiřazena do správné skupiny. Konkrétně, pokud je daný úhel menší než 11°, je zařazen do dané skupiny. Tento úhel byl určen testováním jako nejvhodnější. Vznikal zde však problém, kdy přímka l_i byla skoro rovnoběžná s osou Y. Tehdy byl průsečík s osou Y velmi daleko a tím se výsledné úhly velmi zmenšovaly. Proto zde bylo nutné testovat, zda je úhel menší než 11° a zároveň zda je menší než druhý úhel, jak je znázorněno na obrázku 4.8.



Obrázek 4.8: Úhel α_1 i úhel β_1 jsou menší než 11°, ale úhel α_1 je podstatně menší než úhel β_1.

$$\begin{aligned} l_i &= Ax + By + C \\ P &= (0, -C/B) \\ \vec{v_{1i}} &= (0, Y, 1) \times U_1 \\ \vec{v_{2i}} &= (0, Y, 1) \times U_2 \\ a_1 &= \cos^{-1}(\vec{v_{1i}} \cdot \vec{v_i}) \\ a_2 &= \cos^{-1}(\vec{v_{2i}} \cdot \vec{v_i}) \end{aligned} \tag{4.7}$$

4.4.4 Nalezení dvou hlavních úběžníků scény

Dalším krokem v detekci UMF je nalezení dvou hlavních úběžníků scény. Každý úběžník odpovídá průsečíku všech přímek v dané skupině. Jelikož se nalezené přímky neprotínají v jednom bodě, je nutné použít jiných metod než klasické hledání průsečíku přímek.

Nalezené přímky l_i jsou popsány obecnou rovnicí a odpovídající úběžník v je v homogenních souřadnicích. Poté platí [7]:

$$\forall i : v \cdot l_i = 0 \tag{4.8}$$

Parametry jednotlivých přímek tvoří v reálné projektivní rovině trojrozměrný vektorový prostor bez počátku. Body reálné projektivní roviny korespondují s nadrovinami, které procházejí počátkem. Díky tomu jsme schopni nalézt úběžník velmi přesně a to nasazením nadroviny přes všechny nalezené přímky dané skupiny. Poté normála nadroviny odpovídá hledanému úběžníku v. Normálu nadroviny získáme jako vlastní vektor, který odpovídá nejmenší vlastní hodnotě vypočítané z eigen-dekompozice. Pro výpočet vlastních hodnot

a vektorů pomocí eigen-dekompozice je nutné vytvořit korelační matice C, ze které budou vlastní hodnoty a vektory spočteny [7].

$$C = (l_0 \dots l_N)(l_0 \dots l_N)^T \tag{4.9}$$

Výsledná korelační matice C má rozměr 3 x 3 a je symetrická a díky tomu je eigen-dekompozice velmi efekivní.

Pro výpočet eigen-dekompozice byla opět využita knihovna Eigen, která tuto funkcionalitu poskytuje. Jak již bylo zmíněno, výsledkem jsou vlastní hodnoty a vektory, ve kterých se hledá nejmenší vlastní hodnota a k ní odpovídající vektor. Pro nalezení výsledných souřadnic úběžníku scény je nutné vydělit souřadnice jejich homogenní souřadnicí. Zde však dochází k problému, kdy je úběžník v nekonečnu, tedy homogenní souřadnice je nulová. Tento problém jsme řešili přičtením malé hodnoty k homogenní souřadnici, aby nedocházelo k dělení nulou.

4.4.5 Výpočet hodnot ki+q

Následujícím krokem v detekci je výpočet hodnot ki+q pro každou přímku. Tyto hodnoty se vypočítavají z rovnice 2.3, kde neznámou je právě hodnota ki+q a hodnota b. Zbytek hodnot známe z předchozích kroků algoritmu. Pro výpočet hodnoty ki+q lze tedy využít soustavy rovnic o dvou neznámých. První z nich je již zmíněná hodnota ki+q a druhou je hodnota b, která určuje velikost vektoru, který též není známý.

V této práci jsou hodnoty ki+q vypočteny pomocí metody z práce Ing. Istvána Szentandrásiho [7]. Tato metoda využívá již výše zmíněné soustavy tří rovnic o dvou neznámých. Používá se pseudoinverzní matice a metoda nejmenších čtverců.

Při výpočtu hodnot ki+q je zároveň počítána průměrná hodnota těchto hodnot pro další účely výpočtu, které budou zmíněné později.

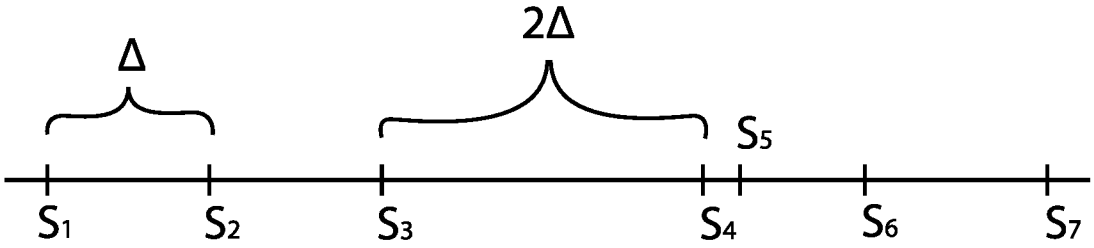
4.4.6 Shlukování

V tomto kroku máme dvě skupiny přímek, které reprezentují hrany v markeru. Avšak jednotlivé hrany jsou reprezentovány více přímkami. Proto je nutné vytvořit shluky přímek, kde každý shluk bude odpovídat jedné hraně v markeru. Tohoto cíle je dosaženo shlukováním přímek dle jejich hodnot ki+q, zjištěných v předchozím kroku. Shlukování se provádí zvlášt pro každou skupinu přímek odpovídající úběžníku scény. Nejprve se provede seřazení přímek dle hodnoty ki+q a poté se prochází přímky jedna po druhé. Aktuální přímka se přidá do shluku a poté se testuje, zdali hodnoty ki+q aktuální a následující přímky se příliš neliší. Pokud ano, je aktuální shluk ukončen, je dopočítán průměr hodnot ki+q v tomto shluku a je vytvořen další shluk. Toto porovnání je prováděno na základě procentuálního rozdílu mezi hodnotami ki+q. Rozdíl hodnot ki+q aktuální l_i a následující přímky l_{i+1} je podělen průměrnou hodnotou kiq_p. Tím získáme procentuální rozdíl těchto hodnot.

$$\frac{(l_i - l_{i+1})}{kiq_p} \tag{4.10}$$

Pokud rozdíl činí více jak 150%, poté následující přímka již patří do dalšího shluku. Přesněji, následující přímka odpovídá další hraně v markeru.

Dále je třeba dopočítat krok Δ mezi jednotlivými shluky. Tímto krokem je myšlena nejčastější vzdálenost mezi shluky. Pro výpočet tohoto kroku je použito opět shlukování, avšak ne dle hodnot ki+q, ale dle průměrných hodnot shluků, vytvořených v předchozí části



Obrázek 4.9: Svislé čáry S_i označují jednotlivé shluky. Mezi těmito shluky je krok δ. Některé shluky mohou chybět a jiné mohou být špatně umístěné jako S_5.

z hodnot ki+q. Jednotlivé shluky jsou procházeny a opět dle pravděpodobnostního rozdílu mezi aktuálním a následujícím shlukem je vytvářena nová skupina shluků. V této skupině jednotlivé shluky obsahují rozdíly průměrných hodnot shluků, vypočteny v první části. A také počet, kolik podobných rozdílů bylo nalezeno. Poté se tyto nově vytvořené shluky procházejí a hledá se maximální počet podobných rozdílů. Z těchto rozdílů je vytvořena průměrná hodnota a je prohlášena za krok mezi shluky.

Na závěr je nutné přiřadit k jednotlivým shlukům správnou hodnotu i, která určuje číslo hrany v markeru. Zde jsme si kladli za cíl, aby hodnota nula ($i = 0$) byla přiřazena prostřednímu shluku. Toho jsme docílili tak, že první hodnota i je nastavena na zápornou hodnotu poloviny počtu shluků. Začíná se od záporných hodnot a tedy hodnota $i = 0$ je vždy přibližně v polovině. Následující hodnoty i jsou dopočítávány jako rozdíl průměrných hodnot shluků podělených krokem mezi shluky, jak je znázorněno v rovnici 4.11.

$$i_{n+1} = i_n + \left(\frac{(ki+q)_{n+1} - (ki+q)_n}{\Delta} \right) \tag{4.11}$$

Díky tomuto výpočtu se vždy přiřadí korektní hodnota i, i když některý shluk chybí.

4.4.7 Lineární regrese

Lineární regrese je matematická metoda, která se používá při proložení množiny bodů přímkou. O jednotlivých bodech lze prohlásit, že jejich x-ové souřadnice jsou víceméně přesné, kdežto y-ové souřadnice mohou být zatíženy chybou. Pokud tyto body proložíme přímkou, vznikne odchylka mezi y-ovou hodnotou na přímce a y-ovou hodnotou bodu. Principem lineární regrese je najít takovou přímku, aby součet druhých mocnin těchto odchylek byl co nejmenší. Hledanou přímku lze vyjádřit rovnicí 4.12, kde se hledají optimální koeficienty k a q.

$$f(i) = ki + q \tag{4.12}$$

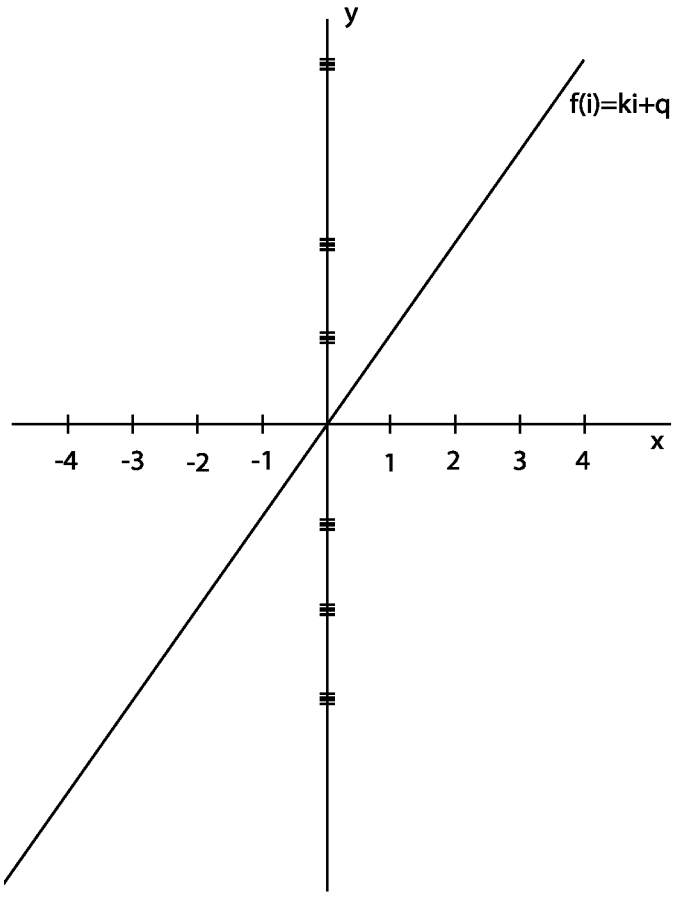
Po získání těchto koeficientů můžeme pouhým dosazením správné hodnoty i do rovnice 2.3 získat přímku reprezentující hranu v markeru.

V této práci se pro výpočet lineární regrese vytvoří dvě množiny hodnot pomocí metody *Calculate*. Jedna množina reprezentuje x-ové souřadnice bodů a druhá y-ové souřadnice bodů. Hodnoty kiq jednotlivých přímek tvoří množinu y-ových souřadnic a množinu x-ových souřadnic tvoří odpovídající hodnoty i vypočítané v předchozím kroku. Z těchto

množin je poté v metodě *slope* dopočítána linerární regrese pomocí metody nejmenších čtverců. Výpočet je založen na následujících rovnicích [14]:

$$\begin{aligned} k &= \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - \left(\sum x_i\right)^2} \\ q &= \frac{\sum x_i^2 \sum y_i - \sum x_i \sum x_i y_i}{n\sum x_i^2 - \left(\sum x_i\right)^2} \end{aligned} \tag{4.13}$$

Pro výpočet těchto rovnic bylo využito knihovny STL jazyka C++. Především pro výpočet jednotlivých sum. Například pro výpočet $\sum x_i$ byla využita funkce *std::accumulate*, která sečte všechny hodnoty v datové struktuře *std::vector*. Pro složené sumy z více položek jako pro $\sum x_i y_i$ byla použita funkce *std::inner_product*, která provede odpovídající výpočet. Následně stačí tyto mezivýpočty dosadit do rovnice a dopočítat koeficienty k a q, které jsou využity v následující části algoritmu.



Obrázek 4.10: Proložení bodů přímkou. Na ose x jsou čísla shluků, na ose y jsou hodnoty ki+q jednotlivých přímek.

4.4.8 Nalezení vějířů

Nyní již máme vše potřebné k tomu, abychom určili vějíře markeru. Pouze stačí využít výsledky předchozích částí algoritmu.Dosadíme do rovnice 2.3 všechny neznámé proměnné,

kde $\hat{I}_base$ je přímka procházející středem obrazu a odpovídajícím úběžníkem scény, $\hat{h}$ je přímka horizontu, hodnota b je rovna 1 a za hodnoty k a q dosadíme výsledek lineární regrese. Poté pouze dosazujeme za hodnoty i celá čísla a výsledkem jsou přímky l_i, kde každá přesně odpovídá jedné hraně v markeru. Tento proces se provede na oba směry přímek a dostaneme kompletní reprezentaci vějířů.

4.4.9 Získání barev středů

Jelikož dosazením celého čísla za hodnotu i dostaneme přímku reprezentující hranu markeru, tak potom hodnotami mezi dvěma celými čísly (1.5) dostaneme přímku, která prochází přesně středem mezi dvěma hranami markeru. Pokud vezmeme dvě přímky, kde každá odpovídá jednomu úběžníku scény, tak jejich průsečík určuje přesně střed jednoho ze čtverců markeru. Tímto způsobem jsme schopni určit středy čtverců markeru s_{ix} a z těchto bodů získáme jejich barvy.

$$\begin{aligned} l_i &= \hat{I}_{base_1} + (k_1 i - q_1)\hat{h} \\ l_x &= \hat{I}_{base_2} + (k_2 x - q_2)\hat{h} \\ s_{ix} &= l_i \times l_x \end{aligned} \tag{4.14}$$

Jelikož osvětlení markeru nemusí být vždy dokonalé, je vhodné získat více barev okolo středu a tyto hodnoty zprůměrovat. V této práci takto průměrujeme barvy okolo středu plus střed samotný. Tím dostáváme lepší hodnotu barvy, než kdybychom brali pouze barvu jednoho bodu.

Při získávání středů čtverců je nutno brát zřetel na směr čtení. Pokud bychom středy čtverců četli jak ve směru tak i proti směru hodinových ručiček, dostali bychom více orientací okna než 4. Řešením je čtení středů čtverců vždy po směru hodinových ručiček. Pro dosažení tohoto cíle stačí provést vektorový součin přímek, kde každá prochází středem obrazu a jedním z úběžníků scény. Pokud je výsledný vektor do obrazu (souřadnice z ¡ 0), tak je třeba v rovnici pro l_i změnit hodnotu k_1 na $-k_1$.

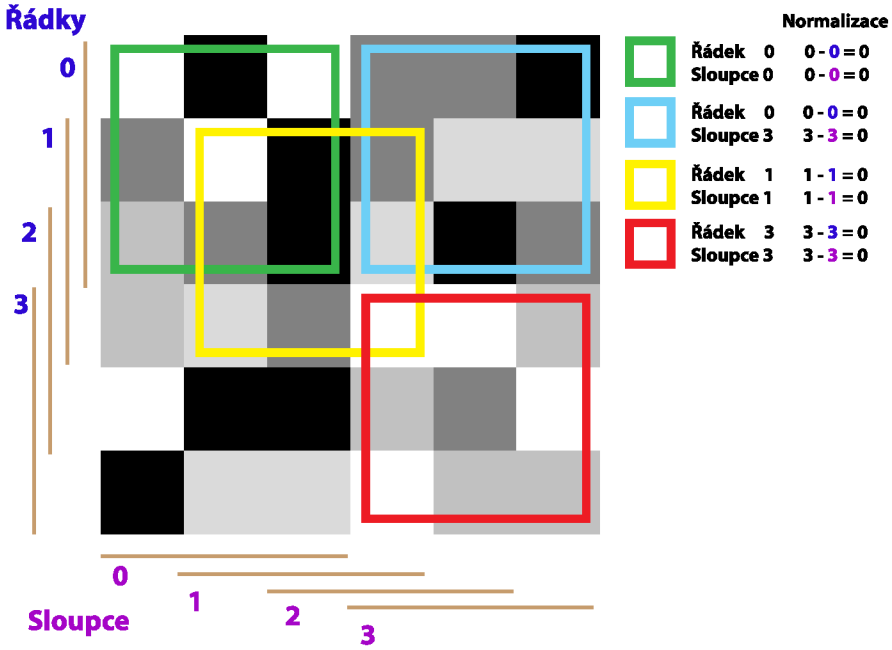
4.4.10 Nalezení pozice a orientace okna

Výsledkem detektoru by měla být pozice a orientace pozorovaného okna v markeru. Tento výsledek jsme nyní schopni vypočítat na základě získaných barev. Porovnáním získaných barev dostaneme reprezentaci okna a tuto reprezentaci potom vyhledáváme v předem vytvořené struktuře. Pokud je okno nalezeno, získáme pozici a orientaci tohoto okna. V této práci však nezískáváme pouze jedno okno, ale detektor získává barvy celkem z 36 bodů. Tedy vybere okno o velikosti 6x6 v okolí středu obrazu. V tomto okně se potom hledá první platně nalezené okno a toto okno je poté prohlášeno za sledované. Hledání prvního platného okna probíhá tak, že se prochází jednotlivá okna, které se nachází v okně 6x6, a porovnávají se barvy tohoto okna. Výsledkem porovnání barev mohou být 3 hodnoty <>=. Aby bylo výsledkem =, musí být rozdíl sousedních barev v rozmezí -20 až 20. Pokud je rozdíl menší jak -20, je výsledkem >, pokud je větší než 20, je výsledkem <. Tímto způsobem se celé okno porovná a poté se hledá pozice a orientace okna.

Před vlastní detekcí je nutné, aby detektor měl načtený marker, ve kterém bude vyhledávat. Toto načtení je možné provést pomocí metody *LoadMarker*, které se předá cesta k souboru s markerem. V této metodě se načítá marker ze souboru a vytvářejí se jednotlivá porovnání. Každé porovnání je poté ukládáno do struktury *unordered_map*. Ke každému

porovnání je také uložena pozice, tedy řádek a sloupec a orientace porovnaného okna v markeru. Pro každé okno je uloženo porovnání všech 4 možných orientací okna. V této struktuře se poté vyhledává a jako klíč se používá právě ono porovnání barev okna.

Jakmile je nalezena pozice a orientace okna, je tato pozice přidána do množiny již nalezených. Pokud se tam však již nachází, je pouze zvýšen počet této pozice a orientace, pokud se v množině nenachází, je nově vložena. Po projití všech oken je ve vytvořené množině pozic hledána ta, která má největší počet výskytu a tato pozice je prohlášena za korektní. Aby však mohlo být využito tohoto principu, bylo nutné, aby pozice, která byla nalezena, byla upravena dle toho, jak jsme se v okně 6x6 posunuli. Na obrázku 4.11 je tento postup znázorněn. Jednotlivá okna jsou procházena a jelikož známe pozici, kde se právě nachazíme,jsme schopni pozice oken normalizovat a určit, která pozice je nejčastější a tu poté prohlásit jako platnou. Na obrázku je vidět, že po normalizaci oken k zelenému oknu, tedy k pozici (0,0), odpovídají všechny ostatní vybraná okna. Může však nastat situace, kdy některému oknu je přiřazena špatná pozice nebo není nalezeno vůbec. Díky tomuto postupu je hledání robustnější a odolnější na chyby.



Obrázek 4.11: Ukázka vyhledání všech oken v okně 6x6 a normalizace pozice.

4.5 Kalibrace kamery

Aby byla pozice a rotace kamery v prostoru korektně spočítána, je nutné nastavit kalibrační matici. Tato matice obsahuje fokální vzdálenost v ose x i v ose y a souřadnice středu obrazu [4].

$$\begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} \tag{4.15}$$

Tuto matici jsme vypočítali na základě série pořízených snímků markeru a za pomoci knihovny OpenCV [4]. Tato knihovna poskytuje funkce pro detekci vyznačných bodů v obraze a funkce pro kalibraci kamery. Výsledky kalibrace pro zařízení Nokia Lumia 920 jsou následovné:

$$\begin{aligned} fx &= 535.7601 \\ fy &= 536.5240 \\ cx &= 315.6434 \\ cy &= 247.3590 \end{aligned} \tag{4.16}$$

Nastavené hodnoty jsou však trošku jiné. A to z toho důvodu, protože DirectX obecně požaduje ideální kameru. Tedy aby fx a fy byly rovny a střed se nacházel ve středu obrazu. Proto námi nastavené parametry jsou následovné:

$$\begin{aligned} fx &= 536 \\ fy &= 536 \\ cx &= 320 \\ cy &= 240 \end{aligned} \tag{4.17}$$

4.6 Vykreslování 3D objektu

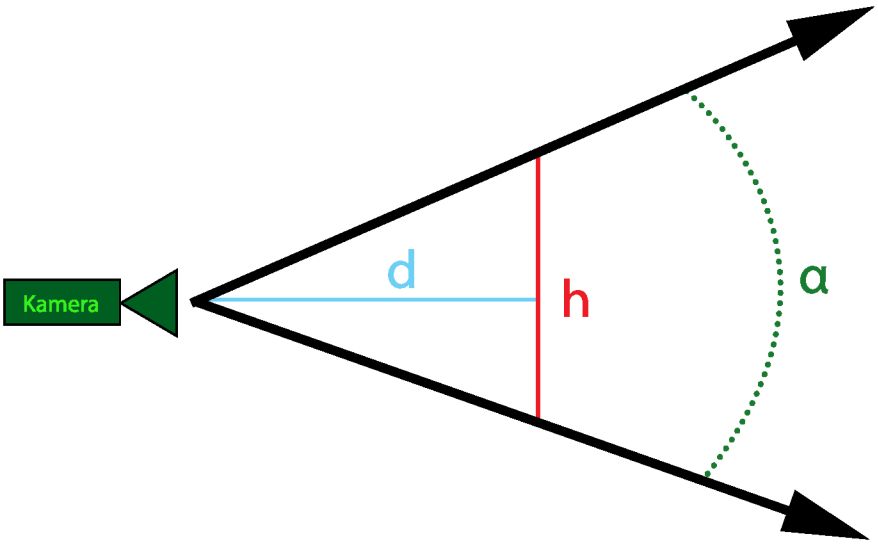
Vykreslování 3D objektu probíhá za pomoci DirectX 3D a UI prvku *DrawingSurface*. Aby tento prvek mohl vykreslovat informace z DX3D vrstvy, je vytvořena instance třídy *Direct3DInterop*, která implementuje potřebná rozhraní. Poté za pomoci metody *CreateContentProvider* v této třídě je nastavena vlastnost *ContentProvider* v UI prvku *DrawingSurface*. Jakmile je toto nastavení provedeno, je vytvořeno vše pro to, aby mohl DirectX korektně vykreslovat objekty do scény. Součástí této inicializace je vytvoření vertexů krychle okolo centrálního bodu, jak je zobrazeno na obrázku 3.4. Také je k těmto vertexům přiřazena barva a je z nich vytvořena krychle.

```
VertexPositionColor cubeVertices[] =
 {
   { XMFLOAT3(-0.5f, -0.5f, -0.5f), cubeColors[0] },
   { XMFLOAT3(-0.5f, -0.5f, 0.5f), cubeColors[0] },
   { XMFLOAT3(-0.5f, 0.5f, -0.5f), cubeColors[0] },
   { XMFLOAT3(-0.5f, 0.5f, 0.5f), cubeColors[0] },

   { XMFLOAT3(-0.5f, -0.5f, 0.5f), cubeColors[1] },
   { XMFLOAT3(0.5f, -0.5f, 0.5f), cubeColors[1] },
   { XMFLOAT3(-0.5f, 0.5f, 0.5f), cubeColors[1] },
   { XMFLOAT3(0.5f, 0.5f, 0.5f), cubeColors[1] },

   { XMFLOAT3(0.5f, -0.5f, 0.5f), cubeColors[2] },
   { XMFLOAT3(0.5f, -0.5f, -0.5f), cubeColors[2] },
   { XMFLOAT3(0.5f, 0.5f, 0.5f), cubeColors[2] },
   { XMFLOAT3(0.5f, 0.5f, -0.5f), cubeColors[2] },

   { XMFLOAT3(0.5f, -0.5f, -0.5f), cubeColors[3] },
   { XMFLOAT3(-0.5f, -0.5f, -0.5f), cubeColors[3] },
   { XMFLOAT3(0.5f, 0.5f, -0.5f), cubeColors[3] },
   { XMFLOAT3(-0.5f, 0.5f, -0.5f), cubeColors[3] },

   { XMFLOAT3(-0.5f, -0.5f, -0.5f), cubeColors[4] },
   { XMFLOAT3(-0.5f, -0.5f, 0.5f), cubeColors[4] },
```

```
    { XMFLOAT3(0.5f, -0.5f, -0.5f), cubeColors[4] },
    { XMFLOAT3(0.5f, -0.5f, 0.5f), cubeColors[4] },

    { XMFLOAT3(-0.5f, 0.5f, -0.5f), cubeColors[5] },
    { XMFLOAT3(-0.5f, 0.5f, 0.5f), cubeColors[5] },
    { XMFLOAT3(0.5f, 0.5f, -0.5f), cubeColors[5] },
    { XMFLOAT3(0.5f, 0.5f, 0.5f), cubeColors[5] }
};
```

Hodnoty vrcholů však nejsou 1 jako na obrázku 3.4, ale poloviční, tak aby délka hrany krychle byla rovna 1. To z toho důvodu, aby při vykreslování odpovídala velikost čtverce markeru vykreslené krychle. Dále je třeba definovat prostor, ve kterém se budou objekty vykreslovat, tzv. frustrum, které je zobrazeno na obrázku 3.6. Toto nastavení se provádí pomocí metody *XMMatrixPerspectiveFovLH*, která potřebuje na vstupu pozorovací úhel kamery v ose y, poměr stran a vzdálenosti k bližšímu a vzdálenějšímu plátnu. Výsledkem je projekční matice. Pozorovací úhel je třeba dopočítat tak, aby odpovídal kalibraci kamery. Tento výpočet je popsán v rovnici 4.18 a zobrazen na obrázku 4.12 [19].

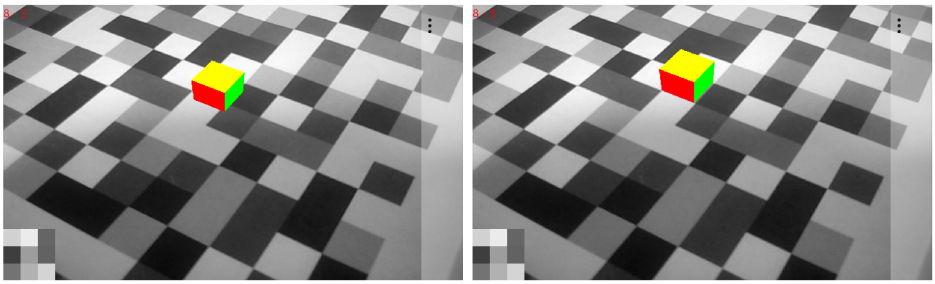
$$\begin{aligned} \tan\left(\frac{\alpha}{2}\right) &= \frac{\frac{h}{2}}{d} = \frac{h}{2d} \\ \frac{\alpha}{2} &= \tan^{-1}\left(\frac{h}{2d}\right) \\ \alpha &= \tan^{-1}\left[\left(\frac{h}{2d}\right)\cdot 2\right] \end{aligned} \tag{4.18}$$



Obrázek 4.12: α = FOV, h = šířka blížšího plátna, d = vzdálenost bližšího plátna od kamery. Inspirováno z [19].

Poté lze již vykreslovat pomocí metody *Render*. Před voláním této metody je vždy ještě volána metoda *Update*, ve které probíhá vytvoření světové a pohledové transformační matice. Světová matice je vytvořena pomocí translace a to pouze v ose y o hodnotu 0.5. Tento posun byl nezbytný, jelikož krychle je definována okolo středového bodu světa a

DirectX polovinu vykresluje pod povrchem a polovina krychle nebyla vidět. Pohledová matice je tvořena na základě vypočítané pozice a rotace kamery. Pomocí translace je z pozice vytvořena translační matice t a z rotačního quaternionu je vytvořena metodou *XMMatrixRotationQuaternion* rotační matice r. Poté jsou tyto matice vynásobeny a vysledek je invertován.



(a) Špatně vykreslený objekt. Objekt nesedí přesně na čtverečku.

(b) Dobře vykreslený objekt. Objekt sedí přesně na čtverečku.

Obrázek 4.13: Ukázka vykreslení objektu bez a s translací v osy y.

```
auto t = XMMatrixTranslation(px, py, pz);
auto r = XMMatrixRotationQuaternion(XMVectorSet(qx, qy, qz, qw))

m_constantBufferData.view = XMMatrixTranspose(XMMatrixInverse(NULL, r*t));
m_constantBufferData.model = XMMatrixTranspose(XMMatrixTranslation(0,0.5,0));
```

4.7 Vykreslování jednotlivých kroků algoritmu

Detektor umožňuje vykreslit jednotlivé kroky detekce přímo do obrazu. Díky tomu je možné vizuálně zobrazit princip celého algoritmu. Toto vykreslování probíhá ve třídě *DrawData*, která obsahuje sadu metod pro vykreslování. Jednou z nich je metoda pro vykreslení přímky z bodu A do bodu B. Toto vykreslení probíhá za pomoci *Bresenham algoritmu*. Detektor obsahuje proměnnou *curstate*, která určuje, v jakém stavu se detektor aktuálně nachází. Poté dle tohoto stavu provádí vykreslování. Tato proměnná může nabývat několika stavů.

- Nothing - nevykresluje se vůbec nic a zároveň se neprovádí ani detekce. V tomto stavu je detektor vypnutý.
- Rozkladové řádky - vykreslují se pouze rozkladové řádky.
- EdgelsPoints - vykreslují se body, které byly detekované na rozkladových řádcích.
- ScanLinesAndEdgelsPoints - vykreslují se jak rozkladové řádky tak i body, které byly detekované.
- Sobel - vykreslují se body, které byly detekované a odhad směru přímek pomocí Sobelova operátoru.
- ScanLinesAndSobel - vykreslují se body, které byly detekované a odhad směru přímek pomocí Sobelova operátoru a rozkladové řádky.

- EdgelsLines - vykreslují se upřesněné přímky.
- RANSAC - vykreslují se obě skupiny přímek, kde každá odpovídá jednomu úběžníku scény. Tyto skupiny jsou barevně odlišeny.
- VanishingPoints - vykreslují se dvě barevně odlišené přímky, které jdou ze středu obrazu do detekovaných úběžníků scény.
- Grid - vykreslují se přímky, které určují detekované vějíře přímek.
- WindowGridCenters - vykreslení středu čtverců detekovaného okna.
- All - vykreslení 3D objektu. V tomto stavu se provádí celý algoritmus.

4.8 Optimalizace rychlosti - NEON

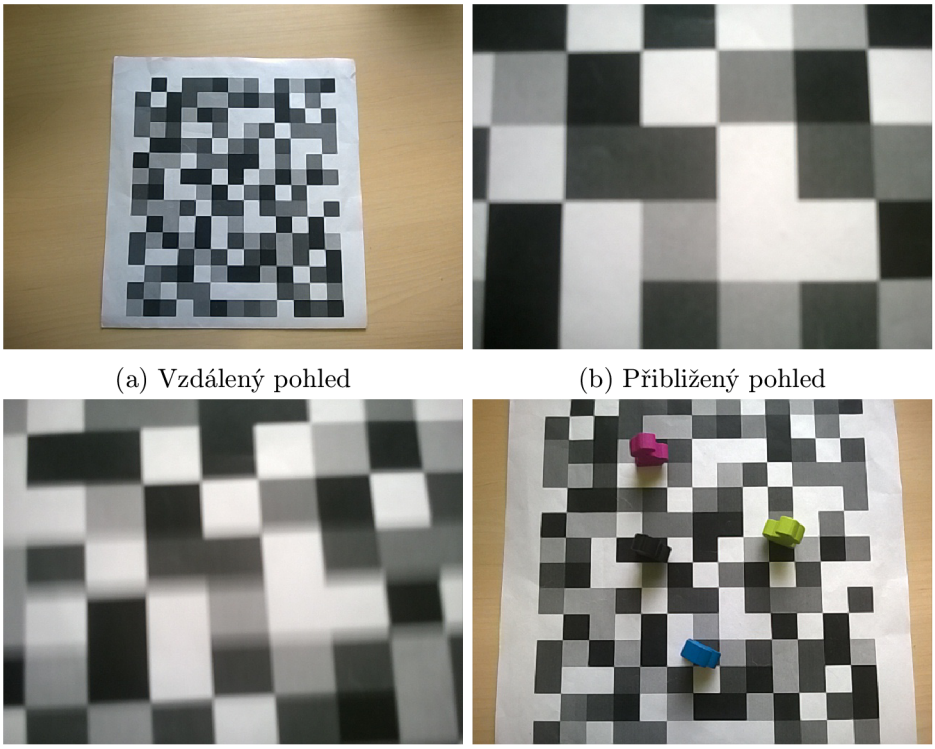
Rychlost detektoru byla optimalizována pomocí NEON jednotky pouze částečně. NEON jednotka byla využita pouze při převodu barevného snímku pořízeného z kamery na černobílý. Avšak v dalších částech nebylo v mých silách využít tuto technologii, jelikož *intrinsics* funkce pracují pouze se základními datovými typy a poli. V této práci jsou využity STL funkce jako *std::vector* apod. Díky tomu jednotlivé cykly nebylo možné urychlit pomocí NEON instrukcí.

4.9 Měření času

Detektor poskytuje informace o časech jednotlivých kroků detekce UMF. Pro měření těchto časů využíváme funkci *QueryPerformanceCounter*, která zprostředkovává přístup k čitači s vysokým rozlišením. Za pomoci této funkce jsou měřeny jednotlivé kroky detekce, kdy vždy před a po kroku detekce získáme aktuální hodnotu čitače a poté z těchto hodnot vypočítáme výsledný čas v milisekundách [16]. Toto měření však probíhá pouze tehdy, pokud je v detektoru povoleno a je složeno ze dvou metod. První z nich je metoda *StartCounter*, která provede vynulování, aby bylo možné počítat od 0. Druhou z metod je *GetCounter*, která vrací počet milisekund od provedení metody *StartCounter*. Tyto hodnoty jsou poté ukládány a předávány v architektuře do vyšších vrstev.

Kapitola 5

Testování



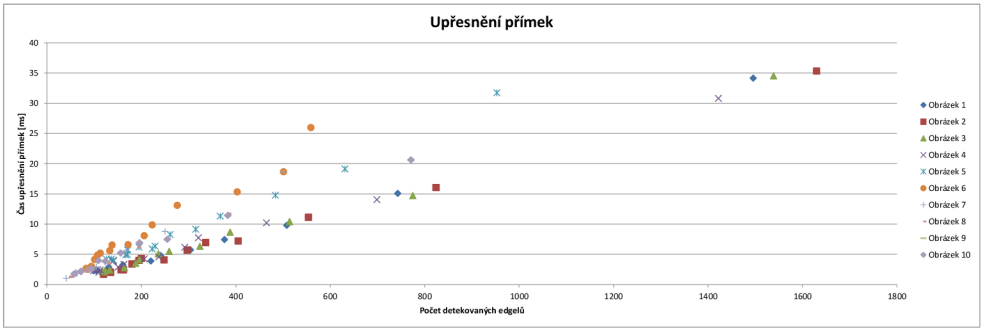
(a) Vzdálený pohled (b) Přibližený pohled

(c) Rozmazaný pohled (d) Pohled zakrytý dalšími objekty

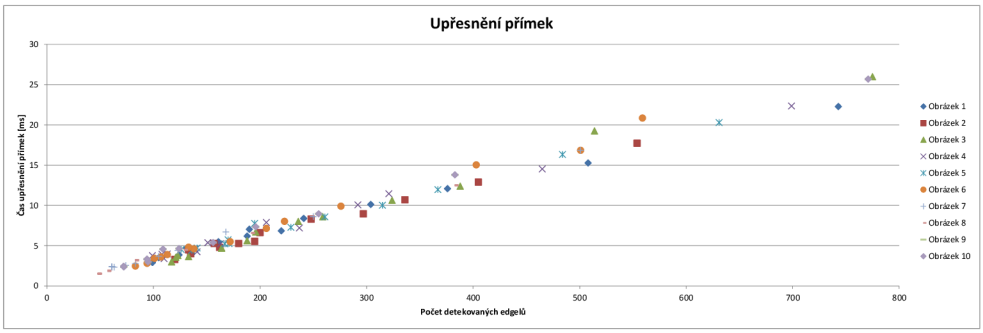
Obrázek 5.1: Ukázka obrázků z testovací sady.

Za účelem testování byla demo aplikace upravena tak, aby nebrala snímky z kamery, ale po spuštění nechala uživatele vybrat fotku z galerie. Díky tomu je možné testy opakovat. Snímky pro testování byly pořízeny v rozlišení 640 x 480 a byly pořízeny v různých úhlech a situacích. Celá sada testovacích snímků je přiložena v příloze B. Testovací snímky obsahují jak pěkně zaostřené, tak i rozostřené a rozmazané snímky, snímky s objekty položenými na markeru i snímky různých vzdáleností od markeru. S každým snímkem byl proveden test, který po otevření snímku postupně zvyšoval offset rozkladových řádků od hodnoty 10 až po 150 pixelů. Algoritmus běžel vždy 5 vteřin a poté byly zaznamenány hodnoty jednotlivých časů detekce, počty detekovaných edgelů a další informace o detekci. Následně byl offset navýšen o 10 pixelů. Testování probíhalo na zařízení Nokia Lumia 920, které

obsahuje dvoujádrový procesor *Qualcomm Snapdragon S4* taktovaný na frekvenci 1.5 GHz a obsahuje 1GB paměti RAM. Pro vykreslování pomocí DirectX je využívána grafická karta *GPU Adreno 225*. Dalším zařízením, na kterém bylo provedeno testování je Nokia Lumia 620. Tento telefon má dvoujádrový procesor *Qualcomm Snapdragon S4* taktovaný na frekvenci 1 GHz a obsahuje 512 MB paměti RAM. Jako grafická karta je zde použita *GPU Adreno 305*. Výsledky testování na těchto zařízeních jsou v příloze C.



(a) Graf upřesnění přímek na zařízení Nokia Lumia 920



(b) Graf upřesnění přímek na zařízení Nokia Lumia 620

Obrázek 5.2: Srovnání upřesnění přímek na zařízeních Nokia Lumia 920 a 620.

Z výsledků výplývá, že při nízkém počtu nalezených edgelů a tím i počtu upřesněných přímek, jednotlivé časy detekce kolísají. To je pravděpodobně způsobeno tím, že algoritmus při nízkém počtu detekovaných edgelů nepracuje v plném rozsahu. V určitém bodě není dostatek dat pro pokračování v algoritmu či jsou data natolik zkreslená, že čas zpracování je naopak velmi dlouhý. Jakmile se počet detekovaných edgelů dostane nad hranici cca 150, poté algoritmus pracuje korektně a časy jsou linárně závislé na počtu detekovaných edgelů. Při testování bylo také zjištěno, že u všech testovacích snímků lze detekovat mřížku markeru a určit pozici a orientaci kamery v prostoru. Avšak u některých toho lze dosáhnout s menším, u jiných naopak s větším offsetem rozkladových řádků. U snímků, které jsou rozmazané a rozostřené, je nutné mít offset mezi 10-40 pixely, jelikož určit hrany a přímky mřížky je v tomto případě náročnější než u zaostřeného snímku. Také objekty položené na markeru, jak je zobrazeno na snímcích 15-20, nepůsobí velké problémy při detekci. To díky tomu, že při upřesňování a rozdělování přímek do dvou hlavních skupin je většina přímek, které neodpovídají markeru úspěšně vyfiltrovány.

Na grafech 5.2a a 5.2b je zajímavé, že na slabším zařízení Nokia Lumia 620 je závislost

času upřesnění přímek na počtu detekovaných edgelů mnohem lineárnější než na zařízení Nokia Lumia 920. Tento jev lze vysvětlit přesností měřící metody. Na zařízení Nokia Lumia 620 jsou časy větší a proto chyba měření a dalších systémových funkcí je zde menší jak u nižších naměřených časů na zařízení Nokia Lumia 920.

V testech bylo dosaženo nejlepšího času detekce *3.05 ms*, při offsetu rozkladových řádků 150 pixelů. Tento výsledek byl dosažen na testovacím snímku 12 a na zařízení Nokia Lumia 920. Bylo dosaženo i nižších časů, avšak při těchto časech nebyla detekce zcela úspěšná. Na zařízení Nokia Lumia 620 bylo dosaženo nejnižšího času *5,32 ms* taktéž na testovacím snímku 12.

Snímek	12
Offset	150
Řádek	7
Sloupec	1
Počet edgelů	78
Počet přímek	36
Detekce edgelů	1,1 ms
Upřesnění přímek	2,3 ms
Nalezení úběžníků	0,2 ms
Výpočet ki+q	0,02 ms
Shlukování	0,1 ms
Lineární regrese	0,1 ms
RANSAC	0,8 ms
Pozice a orientace	0,7 ms
Celkem	5,32 ms

Tabulka 5.1: Časy detekce UMF na zařízení Nokia Lumia 620

Snímek	12
Offset	150
Řádek	8
Sloupec	5
Počet edgelů	78
Počet přímek	36
Detekce edgelů	0,5 ms
Upřesnění přímek	1,2 ms
Nalezení úběžníků	0,1 ms
Výpočet ki+q	0,01 ms
Shlukování	0,1 ms
Lineární regrese	0,04 ms
RANSAC	0,5 ms9
Pozice a orientace	0,6 ms
Celkem	3,05 ms

Tabulka 5.2: Časy detekce UMF na zařízení Nokia Lumia 920

Při testování bylo také ověřeno, že algortimus detekce UMF lze provozovat na dnešních mobilních zařízení s operačním systémem Windows Phone 8 a to v reálném čase i na slabších zařízeních jako je Nokia Lumia 620. Rychlost a úspěšnost zpracovávání snímku záleží na offsetu rozkladových řádků a tím především na počtu detekovaných edgelů a upřesněných přímek.

Kapitola 6

Závěr

Cílem této práce bylo implementovat algoritmus detekce Uniform Marker Fields a demo aplikaci pro Windows Phone 8, která využívá tento algoritmus jako knihovnu. Dalším cílem bylo zjistit, jestli lze provozovat tento algoritmus na slabších zařízeních jako jsou mobilní telefony s operačním systémem Windows Phone 8. Výsledná demo aplikace a knihovna detektoru dokazují, že algoritmus detekce UMF lze provádět na těchto zařízeních.

Při vývoji jsem se setkal s mnoha problémy, které však byly úspěšně vyřešeny a tedy výsledná aplikace je plně provozuschopná. Na testovacích snímcích bylo zjištěno, že detektor je schopen provést uspěšnou detekci i v rozmazaných či objekty překrytých snímcích.

Ale i přesto má algoritmus slabé části. Hlavním problémem je část shlukování, která by si zasloužila zaměnit za robustní shlukovací algoritmus. Také by bylo vhodné refaktorizovat či zcela přepracovat část upřesňování přímek, která z výsledků testů byla nejpomalejší.

Budoucnost této práce vidím ve změně Windows Phone 8 na Windows Phone 8.1, který již umožňuje práci se soubory a složkami a bylo by poté možné načítat různé markery a objekty pro DirectX. Také by v budoucnu bylo vhodné optimalizovat rychlost celého algoritmu pomocí již zmíněné technologie NEON, která v této práci nebyla využita plnohodnotně.

Literatura

[1] ARM: Introducing NEON [online]. 17.3.2010 [cit. 21.5.2014]. Dostupné z: `http://infocenter.arm.com/help/topic/com.arm.doc.dht0002a/DHT0002A_introducing_neon.pdf`.

[2] ARM Connected Community: Coding for NEON - Part 1: Load and Stores [online]. 17.3.2010 [cit. 21.5.2014]. Dostupné z: `http://community.arm.com/groups/processors/blog/2010/03/17/coding-for-neon--part-1-load-and-stores`.

[3] Azuma, R. T.; aj.: A survey of augmented reality. *Presence*, ročník 6, č. 4, 1997: s. 355–385.

[4] Bradski, G.; Kaehler, A.: *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.

[5] Fiala, M.: ARTag, a Fiducial Marker System Using Digital Techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, str. 46.

[6] Guennebaud, G.; Jacob, B.; aj.: Eigen v3 [online]. 8.4.2014 [cit. 21.5.2014]. Dostupné z: `http://eigen.tuxfamily.org`, 2010.

[7] Herout, A.; Szentandrási, I.; Zachariáš, M.; aj.: Five Shades of Grey for Fast and Reliable Camera Pose Estimation. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, s. 1384–1390.

[8] Hirzer, M.: Marker detection for augmented reality applications. *Inst. For Computer Graphics and Vision, Graz University of Technology, Austria*, 2008.

[9] Jared Bienz: Geo AR Toolkit [online]. 25.6.2013 [cit. 21.5.2014]. Dostupné z: `http://gart.codeplex.com`.

[10] Kříž, R.: Uniform Marker Field na válci. 2013.

[11] Kato, H.; Billinghurst, M.; Poupyrev, I.: *ARToolKit*. Washington: Human Interface Technology Laboratory, 2000.

[12] Ľuboslav Lacko: *Vývoj aplikací pro Windows 8.1 a Windows Phone*. Brno: Computer Press, první vydání, 2014.

[13] Lee, H.; Chuvyrov, E.: *Beginning Windows Phone App Development*. Apress, 2012, ISBN 9781430241355.

[14] Meloun, M.; Militký, J.: *Statistická analýza experimentálních dat.* Praha: ACADEMIA, vyd. 2. uprav. rozš. vydání, 2004.

[15] Microsoft: Panorama control for Windows Phone 8 [online]. 10.5.2014 [cit. 21.5.2014]. Dostupné z: `http://msdn.microsoft.com/en-us/ff941104.aspx`.

[16] Microsoft: QueryPerformanceCounter function [online]. 2014 [cit. 21.5.2014]. Dostupné z: `http://msdn.microsoft.com/en-us/ms644904.aspx`.

[17] Microsoft: Visual C++ Language Reference (C++/CX) [online]. 2014 [cit. 21.5.2014]. Dostupné z: `http://msdn.microsoft.com/en-us/hh699871.aspx`.

[18] Microsoft: Windows Phone 8 Reviewer's Guide [online]. 2012 [cit. 21.5.2014]. Dostupné z: `http://download.microsoft.com/download/C/7/A/C7A296B1-D369-479F-A00C-324A87D01AF8/Guide_to_Windows_Phone_8.pdf`.

[19] Miller, T.: *Managed DirectX 9: graphics and game programming: kick start.* Sams Publishing, 2003.

[20] Nokia: C++ support from Windows Phone 8 [online]. 19.1.2013 [cit. 21.5.2014]. Dostupné z: `http://developer.nokia.com/community/wiki/C++_support_from_Windows_Phone_8`.

[21] Nokia: DirectX Developers - DirectX in Windows Phone [online]. 23.8.2012 [cit. 21.5.2014]. Dostupné z: `http://developer.nokia.com/community/wiki/DirectX_Developers_-_DirectX_in_Windows_Phone`.

[22] Nokia: Real-time camera viewfinder filters in Native code [online]. 19.1.2013 [cit. 21.5.2014]. Dostupné z: `http://developer.nokia.com/community/wiki/Real-time_camera_viewfinder_filters_in_Native_code`.

[23] Pokorný, P.: *DirectX-začínáme programovat.* Grada Publishing as, 2008.

[24] Salát, M.: Uniform marker fields pro Android. 2013.

[25] Sherrod, A.: *Beginning DirectX 11 Game Programming.* IT Pro, Course Technology, 2012.

[26] Zuliani, M.: RANSAC for Dummies. *With examples using the RANSAC toolbox for Matlab and more*, 2009.

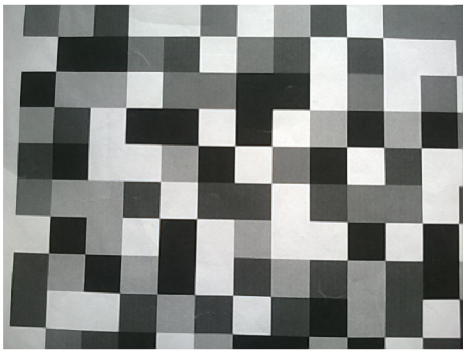
Příloha A

Obsah CD

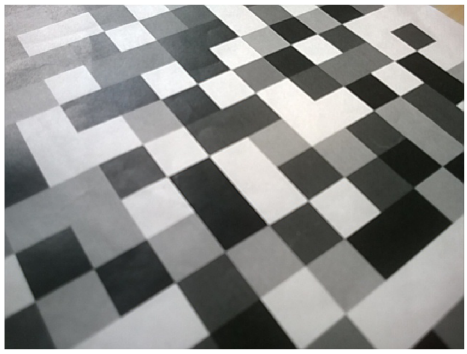
- Zdrojové kódy aplikace
- Výsledná aplikace
- Videoprezentace aplikace
- Technická zpráva
- Zdrojové texty technické zprávy v LaTeX
- Plakát pro prezentaci
- Testovací snímky

Příloha B

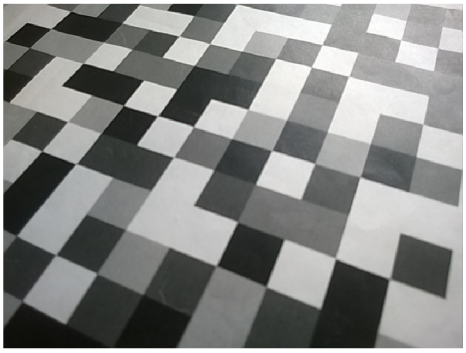
Sada testovacích snímků



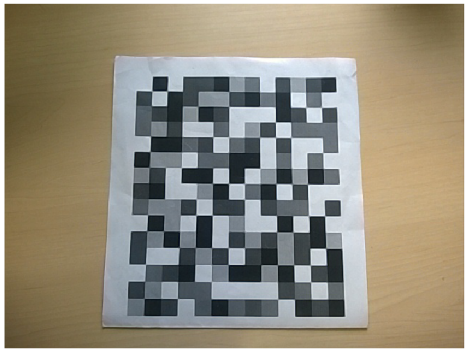
(a) Vzdálený pohled



(b) Přibližený pohled

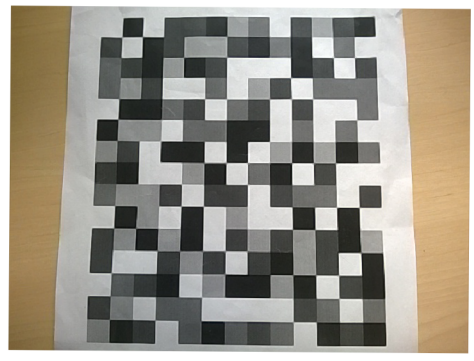


(c) Rozmazaný pohled

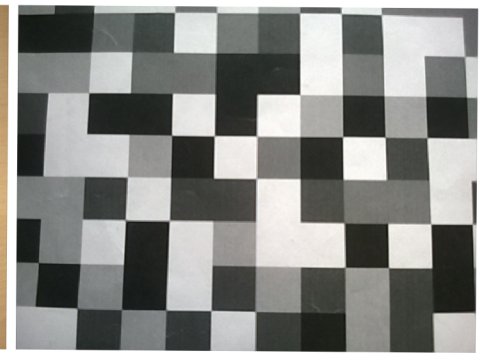


(d) Pohled zakrytý dalšími objekty

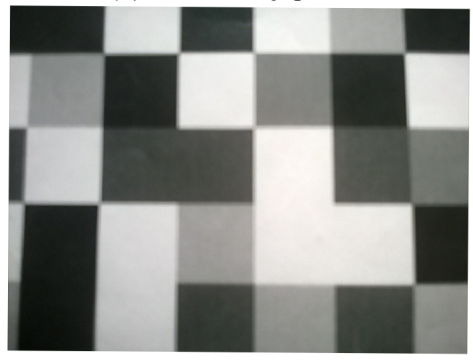
Obrázek B.1: Obrázky 1 - 4 z testovací sady.



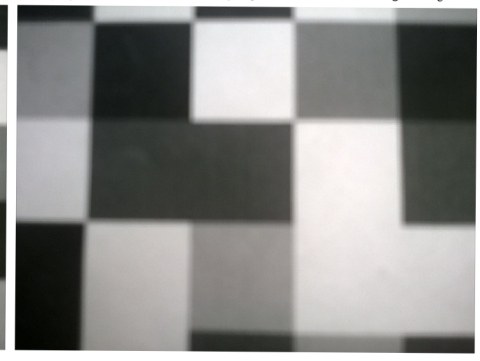
(a) Rozmazaný pohled



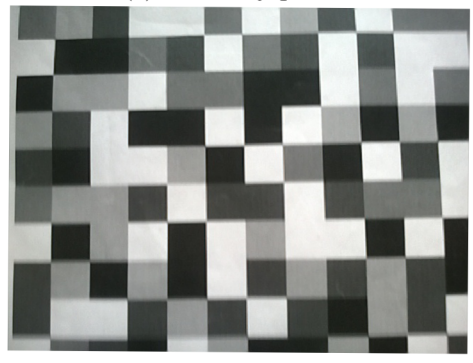
(b) Pohled zakrytý dalšími objekty



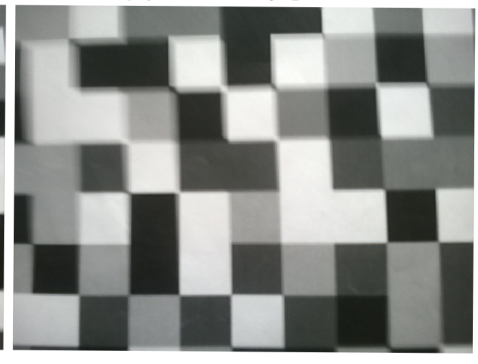
(c) Vzdálený pohled



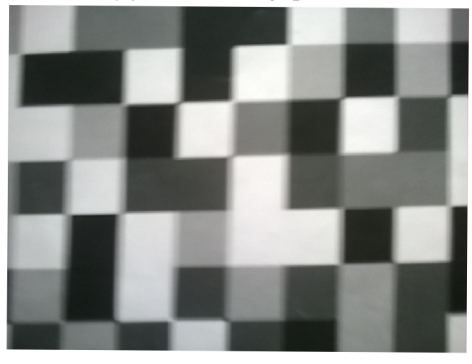
(d) Přibližený pohled



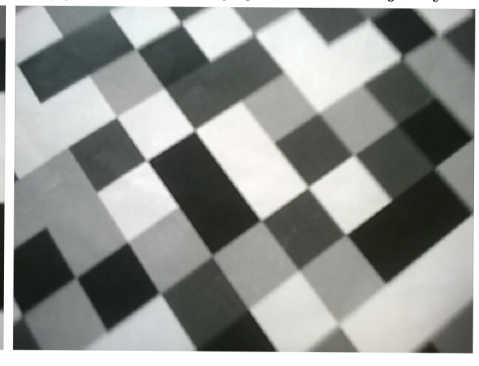
(e) Rozmazaný pohled



(f) Pohled zakrytý dalšími objekty

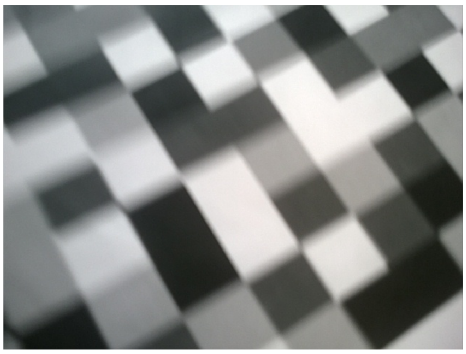


(g) Vzdálený pohled

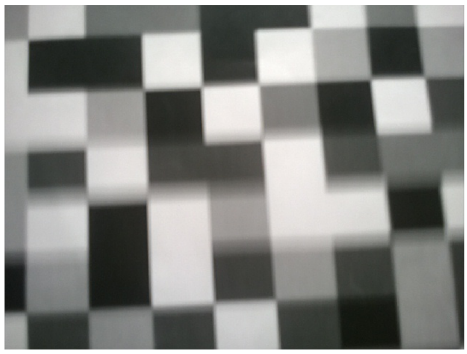


(h) Přibližený pohled

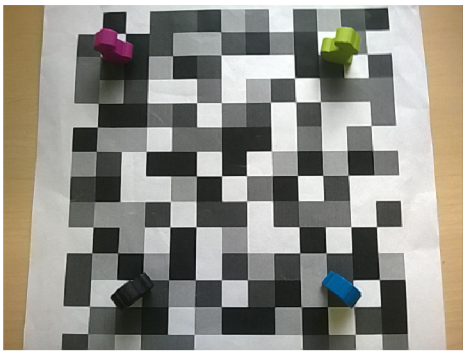
Obrázek B.2: Obrázky 5 - 12 z testovací sady.



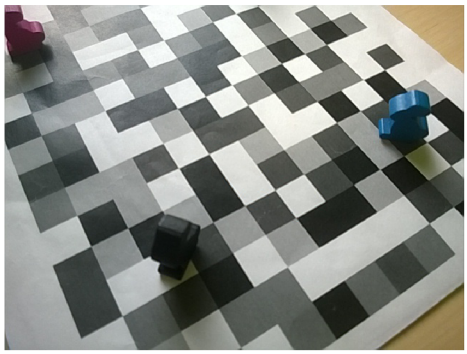
(a) Rozmazaný pohled



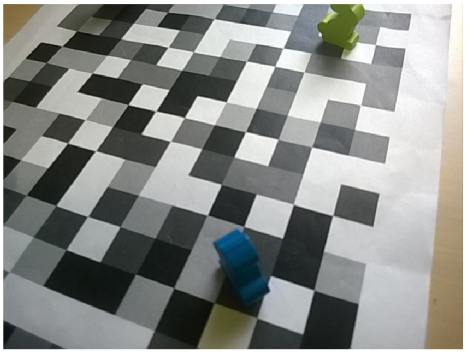
(b) Pohled zakrytý dalšími objekty



(c) Vzdálený pohled

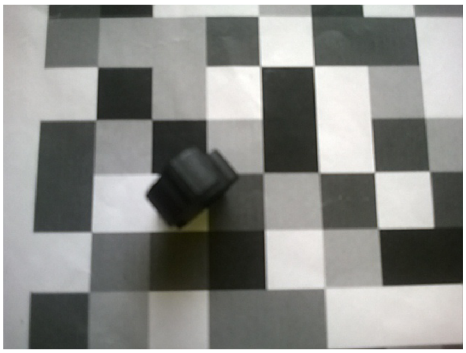


(d) Přiblížený pohled

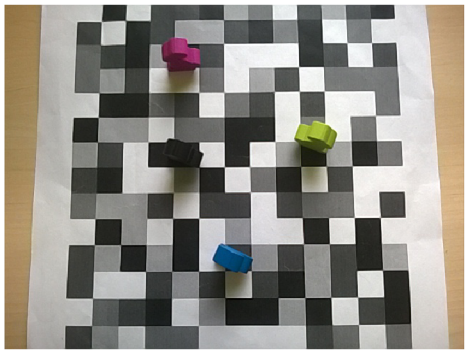


(e) Rozmazaný pohled

(f) Pohled zakrytý dalšími objekty



(g) Vzdálený pohled



(h) Přiblížený pohled

Obrázek B.3: Obrázky 13 - 20 z testovací sady.

Příloha C

Výsledky testování

Tabulka C.1: Výsledky testování na zařízení Nokia Lumia 920

Snímek	Offset	Řádek-Sloupec	Edgely	Přímky	Detekce edgelů	Upřesnění přímek	Uběžníky	Výpočet ki+q	Shlukování	Lin. Regrese	RANSAC	Pozice
Umf01.jpg	10	4r - 3c	1496	881	3,125797531	34,1466963	0,424346	0,449995062	0,7545679	0,257234568	13,45949	0,522904
Umf01.jpg	20	5r - 3c	743	442	1,725728395	15,07498092	0,190325	0,169135802	0,37834792	0,433566779	9,284521	0,481621
Umf01.jpg	30	5r - 3c	508	283	1,524028219	9,79503351	0,153979	0,112917108	0,3042963	0,125499118	3,729862	0,400882
Umf01.jpg	40	5r - 3c	376	214	1,163015432	7,422179012	0,145756	0,08067284	0,25180247	0,118179012	3,199972	0,428151
Umf01.jpg	50	5r - 3c	304	171	0,985695473	5,72036214	0,127698	0,059037037	0,18598628	0,083234568	2,316864	0,456181
Umf01.jpg	60	5r - 3c	241	132	0,858927203	4,776878672	0,115599	0,043862069	0,15117497	0,095851852	2,206868	0,407303
Umf01.jpg	70	5r - 2c	220	117	0,806246465	3,880457912	0,120509	0,032818855	0,11693468	0,074868687	1,294847	0,515359
Umf01.jpg	80	5r - 3c	190	104	0,761074702	3,693496547	0,283234	0,036273697	0,14266416	0,069639674	1,220856	0,520183
Umf01.jpg	90	5r - 2c	188	89	0,90910582	3,609007937	0,12023	0,029478836	0,13306349	0,072079365	1,202386	0,433394
Umf01.jpg	100	5r - 3c	161	85	0,743553001	3,251476373	0,122324	0,029182631	0,15372414	0,069675607	1,206442	0,442513
Umf01.jpg	110	5r - 2c	131	72	0,779579798	2,849333333	0,206801	0,058523906	0,14341818	0,069748148	1,062966	0,552954
Umf01.jpg	120	5r - 2c	124	55	0,668194932	2,172470435	0,116619	0,032818713	0,10272385	0,061190383	0,798786	0,465339
Umf01.jpg	130		105	57	0,660024024	2,272232232	0,121061	0,019083083	0,08001602	0,057245245	1,288765	0,422775
Umf01.jpg	140		99	52	0,702122605	2,265223499	0,140524	0,01715198	0,0816092	0,057657727	0,791563	0,472756
Umf01.jpg	150		99	52	0,702122605	2,265223499	0,140524	0,01715198	0,0816092	0,057657727	0,791563	0,472756
Umf02.jpg	10	5r - 3c	1630	729	5,23773545	35,3462963	0,240116	0,282984127	0,51251852	0,174529101	10,15548	0,499884
Umf02.jpg	20	5r - 3c	824	372	2,525722222	16,04225926	0,1985	0,138606481	0,38769907	0,147625	10,72665	0,509912
Umf02.jpg	30	5r - 3c	554	232	1,590448343	11,11900975	0,186936	0,070697856	0,2022807	0,096311891	3,487587	0,434542
Umf02.jpg	40	6r - 4c	405	167	1,151871176	7,199935588	0,144657	0,057465378	0,17102738	0,086380032	2,337771	0,422435
Umf02.jpg	50	5r - 4c	336	142	1,033390572	6,930356902	0,170731	0,057784512	0,43238721	0,100228956	2,436916	0,43598
Umf02.jpg	60	6r - 4c	297	122	0,970296296	5,698098765	0,154691	0,036197531	0,12579938	0,075015432	1,638735	0,418043
Umf02.jpg	70	7r - 4c	248	102	0,988651852	4,064192593	0,149348	0,03354963	0,11304	0,065019259	1,307523	0,429319
Umf02.jpg	80	6r - 4c	200	79	0,899025641	4,298814815	0,163567	0,029683761	0,14093162	0,11234188	1,224356	0,462299
Umf02.jpg	90	6r - 9c	195	65	0,98818107	3,970979424	0,151329	0,022074074	0,0994856	0,058699588	1,031296	0,654181
Umf02.jpg	100	6r - 4c	180	76	0,887442424	3,388207407	0,131203	0,026243771	0,12355017	0,097546128	1,142141	0,453024
Umf02.jpg	110	5r - 3c	162	60	0,676774603	2,428368254	0,117846	0,014260317	0,0682582	0,043830688	1,144631	0,456631
Umf02.jpg	120	6r - 4c	157	57	0,675366739	2,411230211	0,132845	0,015758896	0,09455047	0,070384895	0,798519	0,416604
Umf02.jpg	130	7r - 4c	133	50	0,618705075	2,078340192	0,148428	0,015838134	0,08691907	0,062156379	0,71144	0,420453
Umf02.jpg	140		135	47	0,600846561	1,990848073	0,134721	0,01681935	0,05976115	0,032789116	0,822104	0,404952
Umf02.jpg	150		120	39	0,815014361	1,693058201	0,11496	0,011540438	0,04211036	0,027313681	0,581663	0,416735
Umf03.jpg	10	7r - 3c	1539	735	3,405142857	34,5255873	0,331386	0,285269841	0,57808466	0,214888889	11,37782	0,450053
Umf03.jpg	20	7r - 4c	775	369	2,461472503	14,71008754	0,198891	0,134936027	0,36319192	0,147232323	7,863354	0,446128
Umf03.jpg	30	7r - 5c	514	240	1,752939476	10,41492683	0,172708	0,094796748	0,29263957	0,105004517	3,735913	0,41523
Umf03.jpg	40	7r - 5c	388	180	1,247001764	8,664751323	0,177326	0,069950617	0,22536155	0,114846561	3,099739	0,436289
Umf03.jpg	50	7r - 4c	324	147	1,005248227	6,32	0,161828	0,052182821	0,17790701	0,095886525	1,991073	0,459448
Umf03.jpg	60	6r - 5c	259	110	0,951580247	5,479993827	0,167191	0,047033951	0,27411728	0,092169753	1,998926	0,531077
Umf03.jpg	70	7r - 5c	236	106	1,03248834	5,067794239	0,16141	0,036203018	0,13651852	0,115799726	1,500524	0,461226
Umf03.jpg	80	7r - 6c	196	85	0,908807553	4,116	0,161993	0,034553377	0,12640813	0,083500363	1,236523	0,466365
Umf03.jpg	90	6r - 5c	188	76	0,822380587	3,490380587	0,148861	0,035931034	0,12598978	0,133440613	1,0526	0,498378
Umf03.jpg	100		164	69	0,707618656	2,79363786	0,133385	0,018356653	0,08271605	0,058513032	0,908422	0,373909
Umf03.jpg	110		164	69	0,707618656	2,79363786	0,133385	0,018356653	0,08271605	0,058513032	0,908422	0,373909
Umf03.jpg	120	7r - 4c	133	48	0,67188694	2,387641326	0,153563	0,015520468	0,09279142	0,05877193	0,745103	0,439454
Umf03.jpg	130	7r - 8c	121	52	0,665995885	2,08982716	0,129156	0,0168107	0,09664198	0,063119342	0,802914	0,493823
Umf03.jpg	140	11r - 9c	123	52	0,669995807	2,326272537	0,146812	0,017903564	0,11172327	0,054015374	0,798396	0,514468
Umf03.jpg	150		117	47	0,57671164	2,138560847	0,145108	0,018314815	0,07236508	0,054407407	0,808016	0,455352
Umf04.jpg	10	5r - 7c	1422	606	4,541692008	30,80261988	0,207096	0,287017544	0,5191501	0,180569201	8,435782	0,379969
Umf04.jpg	20	5r - 6c	699	290	2,198204793	14,04059259	0,196183	0,114021786	0,59904575	0,158413943	5,608244	0,653503
Umf04.jpg	30	4r - 7c	465	190	2,022795821	10,18979297	0,154108	0,07294207	0,27173029	0,099825261	2,938978	0,534477
Umf04.jpg	40	4r - 6c	321	130	1,571234568	7,726818342	0,210215	0,050726631	0,19088183	0,090994709	2,119383	0,523065
Umf04.jpg	50	4r - 7c	292	135	1,158010582	6,155093474	0,140607	0,081502646	0,16990476	0,090902998	1,845873	0,469284
Umf04.jpg	60		237	89	0,940794613	4,538454545	0,132741	0,024319865	0,0917037	0,064760943	1,201182	0,489488
Umf04.jpg	70	9r - 3c	206	101	0,86740388	4,232663139	0,130476	0,033608466	0,11378836	0,06389418	1,734526	0,454956
Umf04.jpg	80		151	54	0,783311728	2,879046296	0,118707	0,015080247	0,06261111	0,049033951	0,759858	0,40162
Umf04.jpg	90		163	69	0,771514212	3,325846684	0,114966	0,025043928	0,08989147	0,05729888	0,902832	0,412217
Umf04.jpg	100		139	64	0,792635391	3,771901235	0,141738	0,021758025	0,07896626	0,058660082	1,087236	0,533574
Umf04.jpg	110		141	53	0,883793881	3,916924316	0,173977	0,022041868	0,10068599	0,051165862	1,012934	0,517478
Umf04.jpg	120		110	35	0,68405475	2,011826087	0,108187	0,009990338	0,03523349	0,029568438	0,667729	0,444605
Umf04.jpg	130		113	46	0,641525253	2,390535354	0,108987	0,016535354	0,0529899	0,048885522	0,649	0,454613
Umf04.jpg	140		108	54	0,601212748	2,432075797	0,112438	0,015393626	0,05572093	0,035548665	0,771025	0,429754
Umf04.jpg	150		99	48	0,898212664	2,270399044	0,11102	0,01325687	0,0410227	0,019044205	0,691622	0,345697

Tabulka C.2: Pokračování výsledků testování na zařízení Nokia Lumia 920

Umf05.jpg	10	4r - 3c	367	165	1,422169182	9,403200732	0,175535	0,073688157	0,27827709	0,144356653	2,844461	0,521626
Umf05.jpg	20	4r - 4c	953	440	2,918148148	31,7097037	0,526815	0,302148148	0,88303704	0,367407407	34,00385	1,511926
Umf05.jpg	30	4r - 4c	631	279	1,926864198	19,12444444	0,233481	0,16508642	0,45817284	0,172691358	6,153531	0,647654
Umf05.jpg	40	4r - 4c	484	229	1,649851852	14,76181481	0,225	0,125518519	0,40192593	0,179	4,731333	0,670593
Umf05.jpg	50	4r - 3c	367	165	1,487654321	11,30882305	0,191984	0,081135802	0,33384362	0,179176955	3,471835	0,621103
Umf05.jpg	60	3r - 10c	315	135	1,217135802	9,136427984	0,177185	0,069975309	0,26824691	0,113514403	2,644412	0,63679
Umf05.jpg	70	4r - 4c	261	134	1,083506173	8,251407407	0,190795	0,064750617	0,29883457	0,160553086	2,654341	0,63519
Umf05.jpg	80		229	109	1,028392593	6,332259259	0,166474	0,054807407	0,2075037	0,090340741	1,908044	0,548993
Umf05.jpg	90	4r - 12c	195	78	0,924592593	6,183481481	0,171407	0,047407407	0,1742963	0,128148148	1,563556	0,594889
Umf05.jpg	100		170	81	0,990255144	4,955061728	0,170782	0,034946502	0,10209053	0,076444444	1,641235	0,613761
Umf05.jpg	110		167	77	0,883676768	4,929023569	0,168956	0,037723906	0,11493603	0,077090909	1,643084	0,620498
Umf05.jpg	120	8r - 3c	171	71	3,680666667	5,708222222	0,175259	0,039037037	0,34355556	0,766740741	1,554815	2,104519
Umf05.jpg	130		136	65	0,786287037	4,194712963	0,17212	0,035916667	0,12487037	0,088351852	1,404565	0,688472
Umf05.jpg	140		126	62	0,715862434	4,182243386	0,160667	0,023746032	0,08625397	0,058296296	1,276571	0,623545
Umf05.jpg	150		141	57	0,712395062	4,011259259	0,181564	0,024971193	0,14495473	0,115144033	1,347901	0,718239
Umf06.jpg	10	4r - 4c	223	139	1,406330084	5,900853801	0,232122	0,048499025	0,17073424	0,130158545	2,160382	0,490264
Umf06.jpg	20	4r - 4c	559	357	3,030666667	25,97303704	0,29763	0,254222222	0,57185185	0,224148148	8,319704	0,700889
Umf06.jpg	30	4r - 4c	403	241	2,131907407	15,33783333	0,252833	0,140611111	0,39011111	0,182111111	6,33337	0,974093
Umf06.jpg	40	4r - 4c	276	164	1,996969697	13,09841077	0,257131	0,119057239	0,32420202	0,147946128	4,431327	0,925279
Umf06.jpg	50	4r - 4c	223	139	1,660424691	9,85677037	0,44883	0,106676543	0,28789136	0,137214815	3,585501	0,784543
Umf06.jpg	60	4r - 4c	206	117	1,956766885	8,079703704	0,22732	0,073986928	0,24485403	0,22843573	2,733098	0,766466
Umf06.jpg	70	4r - 4c	172	103	1,477037037	6,591315697	0,244106	0,138878307	0,23245855	0,122694885	2,406173	0,787175
Umf06.jpg	80	4r - 4c	133	77	1,729674074	5,609792593	0,216659	0,04242963	0,21606667	0,161407407	1,763207	0,802178
Umf06.jpg	90	5r - 4c	138	81	1,686425926	6,534407407	0,21738	0,031342593	0,21016667	0,137259259	2,231194	0,955324
Umf06.jpg	100	4r - 4c	113	72	1,564069136	5,1728	0,191585	0,055634568	0,19382716	0,171792593	1,958212	0,80244
Umf06.jpg	110	5r - 4c	101	50	1,351259259	4,137281481	0,207659	0,032844444	0,20145185	0,187459259	1,619348	0,843474
Umf06.jpg	120		107	49	1,194617284	4,856571429	0,278596	0,025460317	0,11692416	0,104895944	1,466257	0,698173
Umf06.jpg	130		83	41	0,772757202	2,653967078	0,18014	0,020197531	0,08388477	0,058106996	1,043621	0,667407
Umf06.jpg	140	4r - 4c	94	51	0,732407407	2,977696296	0,158007	0,034488889	0,19091852	0,136244444	1,1404	0,677178
Umf06.jpg	150		95	54	0,73178836	2,901063492	0,167349	0,022814815	0,11270899	0,079820106	1,23437	0,619206
Umf07.jpg	10	6r - 7c	501	272	5,178381766	18,66149288	0,298963	0,170461538	0,45737892	0,20822792	6,439658	0,985607
Umf07.jpg	20	6r - 7c	250	136	2,621152263	8,771868313	0,232914	0,124831276	0,24320165	0,109481481	3,057539	0,668576
Umf07.jpg	30	5r - 4c	168	86	1,936518519	5,641809524	0,165302	0,047116402	0,1549418	0,094730159	1,755566	0,600101
Umf07.jpg	40	5r - 4c	123	63	1,339473684	3,700038986	0,148476	0,028038986	0,09674854	0,068717349	1,165033	0,549407
Umf07.jpg	50	5r - 4c	105	51	1,220992993	2,918402402	0,134326	0,018482482	0,07497097	0,064484484	1,038775	0,545882
Umf07.jpg	60		83	41	1,087882353	2,50946841	0,136845	0,020740741	0,07789107	0,070013072	0,841373	0,57156
Umf07.jpg	70	5r - 3c	74	35	1,035442266	2,142313725	0,152863	0,033938998	0,07651852	0,070309368	0,776558	0,572266
Umf07.jpg	80		63	33	0,89932963	1,809444444	0,160615	0,012922222	0,05507037	0,04127037	0,689878	0,51123
Umf07.jpg	90		61	29	0,960148148	1,907039153	0,153672	0,01432381	0,06831323	0,046167196	0,735767	0,623429
Umf07.jpg	100		61	29	0,960148148	1,907039153	0,153672	0,01432381	0,06831323	0,046167196	0,735767	0,623429
Umf07.jpg	110		61	29	0,960148148	1,907039153	0,153672	0,01432381	0,06831323	0,046167196	0,735767	0,623429
Umf07.jpg	120		61	29	0,960148148	1,907039153	0,153672	0,01432381	0,06831323	0,046167196	0,735767	0,623429
Umf07.jpg	130		61	29	0,960148148	1,907039153	0,153672	0,01432381	0,06831323	0,046167196	0,735767	0,623429
Umf07.jpg	140		61	29	0,960148148	1,907039153	0,153672	0,01432381	0,06831323	0,046167196	0,735767	0,623429
Umf07.jpg	150		41	17	0,628777778	1,005432099	0,131395	0,009876543	0,03860494	0,025320988	0,389667	0,679012
Umf08.jpg	10	9r - 3c	383	161	4,087212963	11,72605556	0,183333	0,084342593	0,20738889	0,087435185	3,155759	0,581361
Umf08.jpg	20		192	75	2,156717037	6,158056296	0,13805	0,026008889	0,08624	0,054891852	1,186868	0,58221
Umf08.jpg	30	9r - 3c	129	57	1,603872054	3,486686869	0,123178	0,019723906	0,07850505	0,051030303	0,985556	0,548034
Umf08.jpg	40		94	38	1,123547758	2,479929825	0,124717	0,013769981	0,0625809	0,041808967	0,651181	0,481552
Umf08.jpg	50		83	34	0,990718196	2,086782609	0,169675	0,012334944	0,05096296	0,07773913	0,598982	0,524541
Umf08.jpg	60		83	34	0,990718196	2,086782609	0,169675	0,012334944	0,05096296	0,07773913	0,598982	0,524541
Umf08.jpg	70		57	25	0,994151111	1,593031111	0,197813	0,014405926	0,07248593	0,043638519	0,601025	0,521973
Umf08.jpg	80		57	25	0,994151111	1,593031111	0,197813	0,014405926	0,07248593	0,043638519	0,601025	0,521973
Umf08.jpg	90		48	23	0,862086957	1,237732689	0,128438	0,010022544	0,03945894	0,025610306	0,696477	0,495942
Umf08.jpg	100		48	23	0,862086957	1,237732689	0,128438	0,010022544	0,03945894	0,025610306	0,696477	0,495942
Umf08.jpg	110		48	23	0,862086957	1,237732689	0,128438	0,010022544	0,03945894	0,025610306	0,696477	0,495942
Umf08.jpg	120		48	23	0,862086957	1,237732689	0,128438	0,010022544	0,03945894	0,025610306	0,696477	0,495942
Umf08.jpg	130		48	23	0,862086957	1,237732689	0,128438	0,010022544	0,03945894	0,025610306	0,696477	0,495942
Umf08.jpg	140		48	23	0,862086957	1,237732689	0,128438	0,010022544	0,03945894	0,025610306	0,696477	0,495942
Umf08.jpg	150		48	23	0,862086957	1,237732689	0,128438	0,010022544	0,03945894	0,025610306	0,696477	0,495942
Umf09.jpg	10	5r - 3c	236	136	1,42265755	6,320389744	0,163954	0,05611396	0,1856547	0,121985185	2,135211	0,481787
Umf09.jpg	20	5r - 3c	601	339	2,895718519	20,93287407	0,396756	0,201822222	0,4861037	0,20802963	10,19455	0,978889
Umf09.jpg	30	5r - 3c	396	224	1,928730159	13,05693122	0,293714	0,109301587	0,33754497	0,123597884	6,815026	0,556942
Umf09.jpg	40	5r - 3c	304	159	1,453759259	9,318916667	0,193028	0,115212963	0,25559259	0,153777778	3,136556	0,592741
Umf09.jpg	50	5r - 3c	236	136	1,244313725	7,2456122	0,242675	0,067093682	0,2155817	0,138248366	2,419163	0,596889
Umf09.jpg	60	5r - 3c	192	103	1,197234568	5,633530864	0,161514	0,061061728	0,18023868	0,100279835	1,905119	0,547605
Umf09.jpg	70	4r - 2c	167	88	1,244675926	5,204842593	0,158981	0,034824074	0,16665741	0,153851852	1,700843	0,588093
Umf09.jpg	80	5r - 3c	147	74	1,007992203	4,599321637	0,142175	0,036210526	0,1534347	0,082510721	1,373255	0,638199
Umf09.jpg	90	5r - 3c	147	79	1,005341564	5,441489712	0,148897	0,033621399	0,15106996	0,092773663	1,558099	0,687333
Umf09.jpg	100	5r - 2c	122	64	0,941019608	4,353777778	0,165551	0,028662309	0,1791634	0,113394336	1,550806	0,700558
Umf09.jpg	110	5r - 3c	106	61	1,102953086	3,703111111	0,167664	0,041520988	0,18453333	0,099683951	1,547694	0,778973
Umf09.jpg	120		97	43	0,775521368	2,994096866	0,173265	0,025333333	0,10982336	0,074586895	1,170678	0,706279
Umf09.jpg	130		83	46	0,671423868	2,564378601	0,15372	0,028576132	0,1141893	0,06417284	0,935951	0,560058
Umf09.jpg	140		82	42	0,697636071	2,315942029	0,163369	0,017417069	0,05100161	0,058138486	0,950254	0,569436
Umf09.jpg	150		82	42	0,697636071	2,315942029	0,163369	0,017417069	0,05100161	0,058138486	0,950254	0,569436

Tabulka C.3: Pokračování výsledků testování na zařízení Nokia Lumia 920

Umf10.jpg	10	9r - 3c	771	382	4,208740741	20,61240741	0,358741	0,228111111	5,91996296	0,153222222	30,09174	0,725037
Umf10.jpg	20	9r - 4c	383	195	2,534336182	11,41368661	0,180581	0,081401709	0,2197151	0,096581197	3,764752	0,611214
Umf10.jpg	30	9r - 4c	255	131	1,908464198	7,471861728	0,182262	0,057116049	0,19185185	0,089491358	2,480365	0,639368
Umf10.jpg	40		196	96	1,879654321	6,903518519	0,178654	0,037246914	0,14009877	0,084345679	1,922296	0,687704
Umf10.jpg	50		156	68	1,62477037	5,205037037	0,184163	0,024844444	0,08997037	0,061037037	1,756859	0,616104
Umf10.jpg	60		124	63	1,684987654	3,847703704	0,163901	0,025777778	0,10158025	0,060049383	1,276889	0,63684
Umf10.jpg	70	9r - 3c	109	56	1,115572985	3,958030501	0,177081	0,025830065	0,13305447	0,10248366	1,254126	0,810911
Umf10.jpg	80	6r - 0c	95	40	0,831542088	2,557158249	0,140296	0,01789899	0,08824242	0,056161616	0,870707	0,583434
Umf10.jpg	90		94	40	0,946938272	2,635135802	0,165967	0,019604938	0,09786008	0,058526749	0,893292	0,621465
Umf10.jpg	100		94	40	0,946938272	2,635135802	0,165967	0,019604938	0,09786008	0,058526749	0,893292	0,621465
Umf10.jpg	110		72	25	0,967251029	2,151654321	0,172387	0,016650206	0,16527572	0,048337449	0,743284	0,682667
Umf10.jpg	120		72	25	0,967251029	2,151654321	0,172387	0,016650206	0,16527572	0,048337449	0,743284	0,682667
Umf10.jpg	130		72	25	0,967251029	2,151654321	0,172387	0,016650206	0,16527572	0,048337449	0,743284	0,682667
Umf10.jpg	140		58	30	0,785652422	1,749185185	0,157402	0,012444444	0,05155556	0,01994302	0,716877	0,596957
Umf10.jpg	150		58	30	0,785652422	1,749185185	0,157402	0,012444444	0,05155556	0,01994302	0,716877	0,596957
Umf11.jpg	10	5r - 5c	770	351	4,477806452	17,10508961	0,177529	0,119431302	0,30075508	0,456477897	11,14086	1,011059
Umf11.jpg	20	9r - 4c	389	179	2,019590982	8,133932367	0,143427	0,048273752	0,16807407	0,083958132	2,250332	0,51524
Umf11.jpg	30	9r - 4c	257	115	1,347293932	5,108343578	0,123007	0,029118991	0,10111899	0,073506698	1,811395	0,454736
Umf11.jpg	40	9r - 5c	196	94	1,032717502	4,144598402	0,11652	0,023082062	0,09839361	0,062466231	1,333354	0,446591
Umf11.jpg	50	8r - 1c	157	65	0,992143915	3,0616	0,117896	0,01962328	0,11810794	0,053680423	0,844224	0,442887
Umf11.jpg	60	9r - 10c	133	61	1,143012346	2,637716049	0,106984	0,018930041	0,06044856	0,053193416	1,364642	0,391724
Umf11.jpg	70		112	54	0,912563544	2,396769789	0,125188	0,017676107	0,0661496	0,04421496	0,791495	0,427628
Umf11.jpg	80		93	38	0,689510071	1,579963613	0,111031	0,01119948	0,04502144	0,034245614	0,522622	0,385237
Umf11.jpg	90		96	38	0,72880052	1,730890188	0,127353	0,008790123	0,03729695	0,02325666	0,498745	0,381892
Umf11.jpg	100		96	38	0,72880052	1,730890188	0,127353	0,008790123	0,03729695	0,02325666	0,498745	0,381892
Umf11.jpg	110		96	38	0,72880052	1,730890188	0,127353	0,008790123	0,03729695	0,02325666	0,498745	0,381892
Umf11.jpg	120		70	32	0,660657109	1,169632019	0,104413	0,0100454	0,03684349	0,029577061	0,459766	0,385434
Umf11.jpg	130		70	32	0,660657109	1,169632019	0,104413	0,0100454	0,03684349	0,029577061	0,459766	0,385434
Umf11.jpg	140		70	32	0,660657109	1,169632019	0,104413	0,0100454	0,03684349	0,029577061	0,459766	0,385434
Umf11.jpg	150		70	32	0,660657109	1,169632019	0,104413	0,0100454	0,03684349	0,029577061	0,459766	0,385434
Umf12.jpg	10	7r - 6c	947	521	3,269530864	21,7812037	0,212617	0,180111111	0,31461728	0,138135802	7,616889	0,414512
Umf12.jpg	20	7r - 6c	470	257	1,942793135	9,099992773	0,18383	0,093423668	0,23569286	0,107432701	3,553677	0,404257
Umf12.jpg	30	7r - 6c	324	165	1,38916214	6,054824691	0,183513	0,048697942	0,24645267	0,072510288	2,357844	0,418795
Umf12.jpg	40	7r - 6c	238	122	1,131686275	4,563822803	0,142751	0,037975309	0,1382658	0,059941903	1,962086	0,450004
Umf12.jpg	50	7r - 6c	198	104	0,963954416	3,555327635	0,133536	0,037934473	0,11803419	0,057475783	1,377291	0,437764
Umf12.jpg	60	7r - 6c	165	79	0,942558201	2,936412698	0,130056	0,020497354	0,12316931	0,055746032	1,002947	0,451746
Umf12.jpg	70	7r - 6c	143	70	0,954731952	2,382365348	0,11943	0,023663528	0,25304206	0,054704331	0,843081	0,416188
Umf12.jpg	80	8r - 6c	117	62	0,712877315	2,185826389	0,120653	0,017324074	0,08206713	0,054328704	0,821722	0,40122
Umf12.jpg	90	7r - 12c	112	53	0,711877778	1,989825926	0,118141	0,012937037	0,05526667	0,039533333	0,660926	0,619741
Umf12.jpg	100	7r - 6c	100	51	0,693976068	1,771719658	0,123802	0,015712821	0,07535954	0,047375499	0,772896	0,405187
Umf12.jpg	110	8r - 5c	99	46	0,652011758	1,643136978	0,135208	0,014067019	0,06807055	0,046499706	0,656673	0,427153
Umf12.jpg	120		95	41	0,644301154	1,464432301	0,127055	0,011509411	0,03749848	0,02485003	0,57539	0,411563
Umf12.jpg	130		95	41	0,644301154	1,464432301	0,127055	0,011509411	0,03749848	0,02485003	0,57539	0,411563
Umf12.jpg	140	7r - 5c	74	32	0,581430556	1,275064815	0,122313	0,014555556	0,07379398	0,072518519	0,744685	0,481079
Umf12.jpg	150	8r - 5c	78	36	0,546397306	1,166381594	0,119371	0,013012346	0,05285073	0,044821549	0,49312	0,566958
Umf13.jpg	10	6r - 3c	1111	313	4,777688889	31,21543704	0,184968	0,111308642	0,27297778	0,134795062	5,267388	0,764286
Umf13.jpg	20	6r - 4c	569	145	1,998206349	10,35437037	0,166497	0,043582011	0,16821164	0,089756614	1,94691	0,429185
Umf13.jpg	30	7r - 3c	364	103	2,696560561	6,983971972	0,127447	0,033401401	0,13744945	0,063115115	1,266619	0,433025
Umf13.jpg	40		283	67	3,971543704	6,03778963	0,10941	0,017445926	0,09864889	0,126702222	0,963028	0,407164
Umf13.jpg	50	5r - 3c	226	56	1,160624799	4,560135266	0,116718	0,017916264	0,07923349	0,069597424	0,846277	0,408957
Umf13.jpg	60		183	47	0,76577342	2,833511983	0,12515	0,013102397	0,05389107	0,038923747	0,590414	0,339216
Umf13.jpg	70		164	41	0,772592593	2,787512545	0,107293	0,013471924	0,05176105	0,023832736	1,665806	0,352444
Umf13.jpg	80		164	41	0,772592593	2,787512545	0,107293	0,013471924	0,05176105	0,023832736	1,665806	0,352444
Umf13.jpg	90		133	34	0,711377778	1,977162963	0,112848	0,010177778	0,04452963	0,039648148	0,44317	0,37377
Umf13.jpg	100		124	23	0,635384016	1,842662768	0,118967	0,008912281	0,03148928	0,018120858	0,367481	0,340121
Umf13.jpg	110		113	31	0,66870922	1,737988968	0,10238	0,010846336	0,04741371	0,04294405	0,433734	0,344113
Umf13.jpg	120		113	31	0,66870922	1,737988968	0,10238	0,010846336	0,04741371	0,04294405	0,433734	0,344113
Umf13.jpg	130		113	31	0,66870922	1,737988968	0,10238	0,010846336	0,04741371	0,04294405	0,433734	0,344113
Umf13.jpg	140		89	18	0,587503086	1,280348765	0,114704	0,007688272	0,02838889	0,027185185	0,281657	0,363827
Umf13.jpg	150		89	18	0,587503086	1,280348765	0,114704	0,007688272	0,02838889	0,027185185	0,281657	0,363827
Umf14.jpg	10	4r - 3c	793	332	3,45290535	15,43864198	0,181778	0,10727572	0,25672428	0,108329218	13,57193	0,862132
Umf14.jpg	20	5r - 3c	399	171	1,937936143	10,84236015	0,123806	0,0451341	0,46675862	0,077445722	2,495811	0,451857
Umf14.jpg	30	6r - 7c	263	106	1,264444444	4,793295102	0,114232	0,027541219	0,11905376	0,054685783	2,224913	0,379742
Umf14.jpg	40	14r - 6c	203	99	0,957744325	3,566026284	0,122361	0,073452808	0,09526882	0,063866189	1,109859	0,465806
Umf14.jpg	50	6r - 7c	165	67	1,060128128	2,976668669	0,106042	0,018462462	0,07864264	0,085093093	0,776228	0,386651
Umf14.jpg	60	5r - 4c	138	60	0,783197531	2,449621399	0,111226	0,014148148	0,07386831	0,055547325	0,718979	0,606058
Umf14.jpg	70	11r - 2c	115	43	0,765539452	2,26131401	0,098216	0,011942029	0,05860225	0,034962963	0,629623	0,388754
Umf14.jpg	80		99	39	0,839559259	2,639133333	0,109041	0,009662963	0,03331111	0,015655556	0,603856	0,361115
Umf14.jpg	90		96	33	0,719923924	1,502578579	0,106478	0,01013013	0,03614414	0,021377377	0,479768	0,407343
Umf14.jpg	100		87	36	0,682287582	1,534213508	0,102784	0,013376906	0,06452723	0,042178649	1,208044	0,803664
Umf14.jpg	110		75	30	0,711114556	1,437133506	0,099842	0,010449612	0,03900431	0,032809647	0,426546	0,391283
Umf14.jpg	120		75	30	0,711114556	1,437133506	0,099842	0,010449612	0,03900431	0,032809647	0,426546	0,391283
Umf14.jpg	130		75	30	0,711114556	1,437133506	0,099842	0,010449612	0,03900431	0,032809647	0,426546	0,391283
Umf14.jpg	140		60	18	0,940941176	0,898448802	0,095516	0,006610022	0,02297168	0,014379085	0,269085	0,513407
Umf14.jpg	150		60	18	0,940941176	0,898448802	0,095516	0,006610022	0,02297168	0,014379085	0,269085	0,513407
Umf15.jpg	10	4r - 3c	416	185	1,425828322	8,152287582	0,134226	0,052212636	0,230278	0,094230937	2,750318	0,413956

Tabulka C.4: Pokračování výsledků testování na zařízení Nokia Lumia 920

Umf15.jpg	20	4r - 3c	1007	457	2,048148148	33,25678431	0,214719	0,56951634	0,37342919	0,190640523	10,82118	0,459146
Umf15.jpg	30	5r - 6c	661	277	1,50897037	13,01873333	0,148444	0,080162963	0,28527407	0,099392593	3,565993	0,358489
Umf15.jpg	40	7r - 4c	505	231	1,756177778	9,046251852	0,143662	0,060284444	0,23248	0,091336296	2,969849	0,41779
Umf15.jpg	50	7r - 4c	416	185	1,055792593	8,679146667	0,125369	0,051241481	0,2142637	0,078903704	2,733233	0,588622
Umf15.jpg	60	7r - 4c	320	145	1,163676768	5,9987789	0,117405	0,03538945	0,26124804	0,083191919	1,902101	0,614595
Umf15.jpg	70	6r - 4c	289	124	1,035021164	5,627910053	0,115212	0,032687831	0,26964021	0,078550265	1,420942	0,378788
Umf15.jpg	80		249	103	0,744407407	4,264972222	0,10431	0,022115741	0,11335185	0,071467593	1,131125	0,379486
Umf15.jpg	90		209	82	0,724462402	3,523209877	0,125468	0,023214366	0,10544557	0,072377104	0,978994	0,383663
Umf15.jpg	100	13r - 1c	210	95	0,64982716	3,973015873	0,116713	0,048691358	0,5564515	0,066920635	1,061016	0,632818
Umf15.jpg	110		184	69	0,646061728	3,051641975	0,108012	0,019967078	0,09065432	0,070197531	0,803531	0,372469
Umf15.jpg	120		161	73	0,663272727	2,782671156	0,111178	0,014433221	0,06632099	0,047205387	0,863852	0,383273
Umf15.jpg	130		150	73	1,088893004	2,81462963	0,110905	0,019469136	0,09622222	0,049888889	0,884477	0,415992
Umf15.jpg	140		150	67	0,559986928	2,538496732	0,099874	0,013481481	0,07163834	0,044444444	0,821298	0,387364
Umf15.jpg	150		136	65	0,54437963	2,270615741	0,100639	0,021384259	0,06537963	0,046791667	0,854477	0,334829
Umf16.jpg	10	6r - 1c	511	164	2,070925926	8,78187963	0,146472	0,052386574	0,20402083	0,112018519	2,13316	0,56834
Umf16.jpg	20	6r - 3c	1214	413	2,177787037	38,48361111	0,196741	0,140981481	0,42802778	0,15012037	8,564843	0,562019
Umf16.jpg	30	6r - 2c	818	264	1,746625514	13,70164609	0,178724	0,74508642	1,95628807	1,700921811	9,092156	2,23521
Umf16.jpg	40	6r - 3c	619	206	1,307444444	11,11228395	0,141951	0,05712963	0,67138272	0,084814815	5,616747	0,432136
Umf16.jpg	50	6r - 1c	511	164	2,034598291	8,923834758	0,139311	0,112188034	0,20407977	0,093333333	2,555738	0,411214
Umf16.jpg	60	6r - 3c	408	129	0,895066667	7,936498765	0,126454	0,044953086	0,18320988	0,110819753	1,835057	0,402494
Umf16.jpg	70	7r - 3c	351	110	0,994766667	7,911137037	0,128719	0,027248148	0,15822593	0,0923	1,472989	0,438026
Umf16.jpg	80	0r - 10c	305	98	0,87573591	4,805932367	0,120947	0,020508857	0,09748792	0,073256039	1,430944	0,354673
Umf16.jpg	90	7r - 3c	280	81	0,75945679	5,063160494	0,140247	0,025593715	0,13724355	0,072897868	0,959834	0,416848
Umf16.jpg	100		257	75	0,817239057	4,025113356	0,123066	0,018783389	0,10110887	0,057225589	0,951529	0,4138
Umf16.jpg	110	7r - 2c	246	69	0,641975309	3,738383838	0,115713	0,019510662	0,09944332	0,063407407	0,801172	0,410334
Umf16.jpg	120	7r - 4c	219	69	1,468592593	3,911649832	0,120117	0,016704826	0,1194119	0,05572615	0,851183	0,396462
Umf16.jpg	130	7r - 5c	182	60	0,588570806	2,708736383	0,121102	0,015485839	0,07213508	0,052100218	0,744802	0,392096
Umf16.jpg	140		187	54	0,570736383	2,880013072	0,11061	0,012814815	0,0472854	0,037359477	0,650248	0,394627
Umf16.jpg	150	7r - 6c	198	59	0,543739259	3,312420741	0,110222	0,012699259	0,05450074	0,031911111	0,749908	0,667597
Umf17.jpg	10	8r - 5c	1932	777	3,882281481	40,53342222	0,30317	0,301125926	0,51555556	0,169007407	15,95167	0,473807
Umf17.jpg	20	4r - 6c	980	386	1,713675926	16,37752778	0,198204	0,145824074	0,39606481	0,153203704	27,87979	0,527
Umf17.jpg	30	8r - 6c	627	257	1,892675556	14,89730963	0,163313	0,078198519	0,22115556	0,099004444	3,414827	0,809896
Umf17.jpg	40	8r - 6c	477	181	1,47205168	8,437822567	0,290767	0,05588975	0,19007752	0,080062016	2,902515	0,425092
Umf17.jpg	50	8r - 5c	385	164	1,093465721	6,963180457	0,14898	0,059665879	0,21646651	0,096545311	2,235237	0,465983
Umf17.jpg	60	8r - 6c	315	119	0,889948718	6,778712251	0,156236	0,039350427	0,16949003	0,099529915	1,617501	0,454749
Umf17.jpg	70	7r - 6c	297	105	0,868888889	5,695267806	0,13302	0,029649573	0,16578348	0,070564103	1,336316	0,450493
Umf17.jpg	80	8r - 6c	247	94	0,77083367	4,02263165	0,129376	0,035660606	0,24322963	0,073513805	1,309657	0,526306
Umf17.jpg	90	7r - 6c	231	84	0,880931774	3,599641326	0,126133	0,021122807	0,11322417	0,054795322	0,960943	0,39855
Umf17.jpg	100		202	81	0,670940516	3,07471156	0,128067	0,022891134	0,12626263	0,06982716	0,973859	0,459937
Umf17.jpg	110	7r - 5c	190	74	0,721851852	3,310031746	0,128286	0,019359788	0,08689947	0,065687831	1,301942	0,483714
Umf17.jpg	120		150	50	0,653757106	3,267855297	0,128382	0,01735056	0,05933161	0,045285099	0,638436	0,415435
Umf17.jpg	130		163	56	0,647798519	2,968900741	0,130599	0,016376296	0,06672593	0,046894815	0,81957	0,419804
Umf17.jpg	140		157	48	0,607843466	2,240287911	0,16774	0,019066387	0,07800699	0,0372942	0,602479	0,379829
Umf17.jpg	150		146	59	0,644927407	2,334951111	0,117274	0,017845926	0,08112889	0,049522963	1,018231	0,403147
Umf18.jpg	10		1659	414	4,063222222	29,28946667	0,205533	0,144614815	0,42259259	0,202518519	5,678089	0,439837
Umf18.jpg	20	4r - 3c	840	213	2,389904762	13,56256085	0,159255	0,075420106	0,25107302	0,095851852	3,925672	1,615348
Umf18.jpg	30	6r - 6c	561	137	1,448898551	8,972219002	0,135858	0,042193237	0,1817037	0,075043478	1,773691	0,46677
Umf18.jpg	40		432	104	1,360104019	6,87634673	0,143231	0,033654846	0,10790859	0,061465721	1,207398	0,526827
Umf18.jpg	50		331	85	1,28457716	5,355416667	0,137593	0,027299383	0,13027469	0,062225309	1,133401	0,534657
Umf18.jpg	60		282	64	0,912993939	4,754507744	0,141565	0,015202694	0,06318653	0,073160943	0,940822	0,380251
Umf18.jpg	70		237	63	0,911758945	3,487435028	0,126795	0,019500314	0,09488763	0,051271814	0,872279	0,382195
Umf18.jpg	80	12r - 1c	207	50	0,949240378	3,327067538	0,114739	0,014620189	0,06273638	0,043212781	0,637049	0,444009
Umf18.jpg	90		199	41	0,761125	2,844986111	0,147065	0,012328704	0,06139352	0,038671296	0,570438	0,405211
Umf18.jpg	100		165	45	0,665840629	2,423943883	0,123733	0,014087542	0,06349944	0,044864198	0,68622	0,409605
Umf18.jpg	110		157	35	0,638341158	2,201368805	0,133807	0,009494001	0,03248826	0,024815858	0,512255	0,369817
Umf18.jpg	120		146	30	0,675887831	2,230855026	0,132197	0,010002116	0,04353228	0,028797884	0,416707	0,356607
Umf18.jpg	130		124	27	0,6352	2,196093122	0,124762	0,011352381	0,04907725	0,029111111	0,403892	0,3968
Umf18.jpg	140		108	26	0,595555556	1,714136339	0,122012	0,010988728	0,03726033	0,028809447	0,370929	0,390196
Umf18.jpg	150		115	29	0,589679772	1,77845698	0,121942	0,009356125	0,0405037	0,025196581	0,622591	0,352843
Umf19.jpg	10	8r - 4c	763	423,069	3,434595147	15,31833461	0,399801	0,51051341	0,62246232	0,143167305	11,54334	0,419213
Umf19.jpg	20	8r - 3c	382	208	1,91256503	7,448906115	0,13209	0,243248923	0,16800345	0,112413437	2,700038	0,390033
Umf19.jpg	30	8r - 3c	254	136	1,345998488	4,622086168	0,12302	0,045768707	0,12266364	0,066040816	2,239335	0,42094
Umf19.jpg	40	8r - 4c	199	106	1,174648889	3,700613333	0,122604	0,039754074	0,12405333	0,173274074	1,58789	0,519241
Umf19.jpg	50	8r - 4c	162	87	0,957320106	3,323690476	0,117698	0,028880952	0,10914286	0,060915344	0,974172	0,476492
Umf19.jpg	60	8r - 3c	131	67	0,788461445	2,357850638	0,110812	0,026775956	0,08710625	0,05636187	0,905622	0,412964
Umf19.jpg	70	8r - 4c	106	58	0,760282187	2,072213992	0,098257	0,021370958	0,08043269	0,049895356	0,765768	0,711283
Umf19.jpg	80		91	49	0,635624143	1,785615912	0,094294	0,013802469	0,05424143	0,040831276	0,590875	0,325473
Umf19.jpg	90		92	45	0,635917379	1,656367521	0,10961	0,013529915	0,05497721	0,036692308	0,564362	0,384339
Umf19.jpg	100		80	43	0,674547619	1,443396825	0,106915	0,011666667	0,05387037	0,039862434	0,553116	0,36396
Umf19.jpg	110		71	38	0,595794239	1,229920439	0,106554	0,01363786	0,05250206	0,035303155	0,565882	0,438494
Umf19.jpg	120		69	32	0,595116402	1,24768254	0,100021	0,009846561	0,03972751	0,032843915	0,431601	0,358508
Umf19.jpg	130		60	27	0,563530446	1,217042059	0,102621	0,010028876	0,03743377	0,027430006	0,385484	0,34855
Umf19.jpg	140		52	29	0,687521767	1,006853801	0,09292	0,01300065	0,04698895	0,035547758	0,428564	0,509949
Umf19.jpg	150		52	29	0,687521767	1,006853801	0,09292	0,01300065	0,04698895	0,035547758	0,428564	0,509949

Tabulka C.5: Pokračování výsledků testování na zařízení Nokia Lumia 920

Umf20.jpg	10	4r - 3c	1954	876	3,889796296	41,80822222	0,273389	0,347222222	0,65474074	0,214037037	12,61548	0,413333
Umf20.jpg	20	4r - 3c	975	425	1,938573388	19,59453498	0,203759	0,179829904	0,48528944	0,158134431	10,92436	0,421822
Umf20.jpg	30	4r - 4c	640	281	1,842607407	12,84979894	0,151653	0,090835979	0,28804233	0,10437672	3,961266	0,435458
Umf20.jpg	40	4r - 3c	475	207	1,337637037	9,203603704	0,136078	0,084066667	0,23863704	0,117777778	2,847622	0,425007
Umf20.jpg	50	5r - 4c	404	171	0,958941283	7,945528455	0,166869	0,058485998	0,20774345	0,100224029	2,436264	0,460784
Umf20.jpg	60	4r - 4c	315	139	0,890740741	6,151828283	0,120505	0,043309764	0,19338721	0,282956229	2,015451	0,614751
Umf20.jpg	70	4r - 4c	272	113	0,832410494	4,890209877	0,111253	0,041027778	0,17281481	0,113166667	1,295438	0,370272
Umf20.jpg	80	5r - 3c	219	89	0,746906225	4,419911742	0,120296	0,031114263	0,15567218	0,096630418	1,105765	0,427868
Umf20.jpg	90		230	94	0,776314815	4,601958333	0,156148	0,039375	0,24433796	0,082032407	1,284611	0,506796
Umf20.jpg	100		207	87	0,768651235	4,049580247	0,162954	0,031660494	0,13693519	0,075888889	1,159361	0,398843
Umf20.jpg	110		197	76	0,735976955	3,602251852	0,12135	0,023535802	0,10179424	0,05782716	1,058318	0,436602
Umf20.jpg	120		158	67	0,672553086	3,071045267	0,117205	0,021129218	0,0693037	0,049093004	0,960965	0,414074
Umf20.jpg	130		156	71	0,989114638	3,305022928	0,114586	0,021181658	0,09887831	0,05945679	1,060004	0,441929
Umf20.jpg	140		140	58	0,570965657	2,527307744	0,119763	0,024180471	0,07488215	0,061322559	0,78702	0,392913
Umf20.jpg	150		137	58	0,553175729	2,831505122	0,129289	0,019211978	0,07209771	0,048803783	0,896785	0,463584

Tabulka C.6: Výsledky testování na zařízení Nokia Lumia 620

Snímek	Offset	Řádek-Sloupec	Edgely	Přímky	Detekce edgelů	Upřesnění přímek	Uběžníky	Výpočet ki+q	Shlukování	Lin. Regrese	RANSAC	Pozice
Umf01.jpg	10	4r - 3c	1496	881	5,124079772	45,71008547	0,395932	0,55819943	0,83895157	0,288649573	15,65707	0,510815
Umf01.jpg	20	5r - 3c	743	442	3,320071958	22,26429206	0,362565	0,256575661	0,61549206	0,28768254	10,59751	0,581147
Umf01.jpg	30	5r - 3c	508	283	2,701642512	15,25771659	0,257069	0,165091787	0,40639614	0,168038647	5,755816	0,536126
Umf01.jpg	40	5r - 3c	376	214	1,918249433	12,07926531	0,219277	0,114325019	0,26647317	0,168072562	4,22039	0,55298
Umf01.jpg	50	5r - 3c	304	171	2,083813757	10,10777566	0,22306	0,120160847	0,33128466	0,132888889	3,987903	0,661096
Umf01.jpg	60	5r - 3c	241	132	1,701118908	8,369216374	0,231778	0,125840156	0,32831579	0,125387914	2,869072	0,595938
Umf01.jpg	70	5r - 2c	220	117	1,439477778	6,835814815	0,175778	0,053348148	0,18947407	0,091751852	2,31673	0,560078
Umf01.jpg	80	5r - 3c	190	104	1,386603704	7,020103704	0,246804	0,071014815	0,21474815	0,114203704	2,371422	0,635978
Umf01.jpg	90	5r - 2c	188	89	1,484497626	6,183073124	0,180084	0,044296296	0,18388604	0,093409307	1,837109	0,620015
Umf01.jpg	100	5r - 3c	161	85	1,321442681	5,476585538	0,211312	0,048007055	0,18563668	0,089368607	1,785771	0,615344
Umf01.jpg	110	5r - 2c	131	72	1,621957672	4,669848325	0,176078	0,04810582	0,22310758	0,086532628	1,479499	0,595665
Umf01.jpg	120	5r - 2c	124	55	1,107461608	3,847082204	0,217604	0,032657633	0,18249684	0,095523035	1,256578	0,652173
Umf01.jpg	130		105	57	1,12256229	3,504188552	0,180158	0,025265993	0,21444781	0,084249158	1,128889	0,597559
Umf01.jpg	140		99	52	1,082844444	2,91858107	0,156458	0,070492181	0,09285926	0,076915226	1,133047	0,681123
Umf01.jpg	150		99	52	1,082844444	2,91858107	0,156458	0,070492181	0,09285926	0,076915226	1,133047	0,681123
Umf02.jpg	10	6r - 7c	1630	729	10,50248148	55,48222222	0,406889	0,541333333	1,12148148	0,321	16,55485	0,673481
Umf02.jpg	20	6r - 7c	824	372	3,38821164	24,83326984	0,344233	0,321121693	0,59648677	0,230455026	19,41568	0,664614
Umf02.jpg	30	6r - 7c	554	232	3,196833333	17,71625926	0,251623	0,113697531	0,36650617	0,137259259	5,910204	0,603062
Umf02.jpg	40	5r - 7c	405	167	2,152467236	12,88602279	0,672712	0,092245014	0,29464387	0,113931624	4,172268	0,577977
Umf02.jpg	50	6r - 7c	336	142	2,130958184	10,69307527	0,297434	0,271612903	0,24728315	0,154121864	3,502151	0,633176
Umf02.jpg	60	5r - 6c	297	122	1,600080423	8,949468783	0,214624	0,058797884	0,19181376	0,138268783	2,60345	0,731886
Umf02.jpg	70	7r - 7c	248	102	2,452318083	8,288522876	0,230932	0,046126362	0,1999695	0,101690632	2,224902	0,593316
Umf02.jpg	80	6r - 7c	200	79	1,478358358	6,609093093	0,230442	0,051079079	0,16617417	0,117189189	1,763692	0,760276
Umf02.jpg	90	6r - 6c	195	65	1,254201646	5,536238683	0,233144	0,028300412	0,19467901	0,094654321	1,711128	0,692399
Umf02.jpg	100	5r - 7c	180	76	1,336125356	5,254803419	0,195225	0,043297246	0,13538082	0,113777778	1,528809	0,622264
Umf02.jpg	110		162	60	1,770670465	4,794222222	0,230978	0,022666667	0,10026591	0,070222222	1,229633	0,61475
Umf02.jpg	120	6r - 7c	157	57	1,220966862	5,263929825	0,193485	0,217808967	0,14546199	0,076136452	1,439942	0,721329
Umf02.jpg	130	6r - 6c	133	50	1,282311111	4,458388889	0,241967	0,024388889	0,22574444	0,07522963	1,107407	0,751233
Umf02.jpg	140		135	47	1,499888889	3,983614815	0,205381	0,026985185	0,07853333	0,054733333	1,198615	0,920956
Umf02.jpg	150		120	39	1,534951852	3,2767	0,18977	0,015537037	0,06285926	0,036377778	0,812719	0,566096
Umf03.jpg	10	7r - 3c	1539	735	6,93026455	55,58549962	0,554029	0,561034014	1,0808254	0,407071807	19,59475	0,860218
Umf03.jpg	20	7r - 4c	775	369	3,325227513	25,96956614	0,771598	0,25675485	0,91255732	0,260246914	11,98604	0,953481
Umf03.jpg	30	7r - 5c	514	240	2,723259259	19,23077641	0,384905	0,27618107	0,57587929	0,247566529	6,540477	0,823259
Umf03.jpg	40	7r - 5c	388	180	2,282153086	12,37253827	0,265422	0,089471605	0,33736296	0,173101235	4,672049	0,675254
Umf03.jpg	50	7r - 4c	324	147	2,118310185	10,61639815	0,256866	0,090259259	0,32871296	0,171435185	3,290148	0,818852
Umf03.jpg	60	6r - 5c	259	110	1,663152263	8,580436214	0,243564	0,060292181	0,19820576	0,125502058	2,609798	0,70684
Umf03.jpg	70	7r - 5c	236	106	1,508318083	8,004396514	0,205791	0,047494553	0,22613508	0,136165577	2,383856	0,659786
Umf03.jpg	80	7r - 6c	196	85	1,849654321	6,680341564	0,229749	0,040695473	0,16239095	0,095061728	1,742967	0,598802
Umf03.jpg	90	6r - 5c	188	76	1,604078189	5,596123457	0,52549	0,0401893	0,17133745	0,13055144	1,490922	0,723626
Umf03.jpg	100		164	69	1,439435435	4,675583584	0,213129	0,049197197	0,12211411	0,083623624	1,446823	0,915159
Umf03.jpg	110		164	69	1,439435435	4,675583584	0,213129	0,049197197	0,12211411	0,083623624	1,446823	0,915159
Umf03.jpg	120	7r - 4c	133	48	1,161675214	3,619525166	0,178401	0,021941121	0,08867996	0,098362773	1,047221	0,629246
Umf03.jpg	130	7r - 8c	121	52	1,141025926	3,548103704	0,213567	0,022088889	0,09592963	0,079455556	1,412481	0,765044
Umf03.jpg	140	11r - 9c	123	52	1,41194152	3,776810916	0,208409	0,021883041	0,1771345	0,095692008	1,131634	0,721084
Umf03.jpg	150		117	47	1,119350427	2,98679962	0,23423	0,024733143	0,10675404	0,058264008	1,223088	0,763487
Umf04.jpg	10	4r - 3c	1422	606	5,527489712	47,38301235	0,38884	0,523555556	0,83290535	0,271374486	13,95682	0,545202
Umf04.jpg	20	4r - 4c	699	290	3,123950617	22,33675309	0,557241	0,138141975	0,62750617	0,25862963	7,895432	0,975685
Umf04.jpg	30	4r - 3c	465	190	2,342084291	14,51216858	0,259367	0,086687101	0,43146871	0,144904215	4,575045	0,735581
Umf04.jpg	40	5r - 5c	321	130	2,330027778	11,43968981	0,218	0,051509259	0,22455556	0,17525	2,864847	0,764958
Umf04.jpg	50	12r - 6c	292	135	1,936938272	10,06357351	0,208314	0,066891134	0,29607632	0,110617284	2,914635	0,627847
Umf04.jpg	60		237	89	1,631504762	7,197680423	0,206535	0,027957672	0,15853545	0,082738624	1,811407	0,625956
Umf04.jpg	70	9r - 12c	206	101	1,993407407	7,874522876	0,184885	0,035529412	0,15869281	0,090379085	1,934819	0,71983
Umf04.jpg	80		151	54	1,551868313	5,36144856	0,186527	0,035674897	0,09776132	0,056802469	1,296123	0,692247
Umf04.jpg	90	10r - 3c	163	69	1,415246561	5,228393651	0,215924	0,027602116	0,19742646	0,089955556	1,523759	0,775022
Umf04.jpg	100		139	64	1,614967078	4,571082305	0,170506	0,034028807	0,09117284	0,051897119	1,353996	0,650313
Umf04.jpg	110		141	53	1,250898386	4,227149098	0,225561	0,027297246	0,10747958	0,061409307	1,122526	0,617804
Umf04.jpg	120		110	35	1,16817094	3,372007597	0,181717	0,011912631	0,05696106	0,032364672	0,838241	0,659662
Umf04.jpg	130		113	46	1,307543544	3,950086086	0,175195	0,019543544	0,08887688	0,046790791	1,093397	0,579688
Umf04.jpg	140		108	54	1,408465021	3,91644856	0,168267	0,041061728	0,07590123	0,06727572	1,52907	0,568708
Umf04.jpg	150		99	48	1,100709552	3,751329435	0,178643	0,043122807	0,07951657	0,050487329	1,331177	0,716733
Umf05.jpg	10	4r - 3c	1870	867	7,825182834	66,58989536	0,543374	0,603673133	1,15801999	0,40122281	22,10713	0,593277
Umf05.jpg	20	4r - 4c	953	440	3,213542484	33,97093682	0,450824	0,275398693	1,04687582	0,368061002	13,25244	1,225813
Umf05.jpg	30	4r - 4c	631	279	2,543442963	20,27915259	0,318957	0,16477037	0,45096296	0,191235556	6,816729	0,66765
Umf05.jpg	40	4r - 4c	484	229	2,330873016	16,31722222	0,394582	0,120952381	0,49719577	0,226248677	5,050106	0,644143
Umf05.jpg	50	4r - 3c	367	165	1,976215054	11,94651374	0,242552	0,110738351	0,35146476	0,157667861	4,04745	0,65395
Umf05.jpg	60	3r - 10c	315	135	1,904430556	10,01581944	0,219449	0,068481481	0,30478241	0,100578704	2,847472	0,614222
Umf05.jpg	70	4r - 4c	261	134	1,630617284	8,559735129	0,304956	0,089867565	0,28175982	0,152139169	3,169158	0,734281
Umf05.jpg	80		229	109	1,840989107	7,257215686	0,232475	0,064291939	0,23730283	0,111694989	2,130392	0,570292
Umf05.jpg	90	4r - 12c	195	78	1,497354497	7,75931746	0,218349	0,043153439	0,24385185	0,102449735	2,113889	0,604683
Umf05.jpg	100		170	81	1,715139918	5,721753086	0,208901	0,032506173	0,10561728	0,118057613	1,66535	0,573704
Umf05.jpg	110		167	77	1,321600823	5,216234568	0,176868	0,037707819	0,12544856	0,074028807	1,633852	0,54128
Umf05.jpg	120	8r - 3c	171	71	1,572516129	5,26153405	0,184817	0,048033453	0,17922581	0,120291517	1,445281	0,618695
Umf05.jpg	130		136	65	1,103356725	4,400510721	0,194963	0,031582846	0,11710331	0,107364522	1,313275	0,842121
Umf05.jpg	140		126	62	1,230654971	4,520526316	0,152838	0,026826511	0,61099805	0,049134503	1,371306	0,588581
Umf05.jpg	150		141	57	1,284477366	4,697555556	0,178733	0,02255144	0,12123045	0,096868313	1,229074	0,563481

Tabulka C.7: Pokračování výsledků testování na zařízení Nokia Lumia 620

Umf06.jpg	10	4r - 4c	1100	696	5,992719577	41,80761905	0,492487	0,418328042	0,67771429	0,246539683	18,3197	0,59964
Umf06.jpg	20	4r - 4c	559	357	3,027394525	20,84569404	0,370937	0,366866345	1,35408696	0,266982287	8,525655	1,646692
Umf06.jpg	30	4r - 4c	403	241	2,834502646	15,0274709	0,257471	0,132677249	0,46839683	0,202084656	6,022164	0,667979
Umf06.jpg	40	4r - 4c	276	164	2,361407407	9,887871605	0,234054	0,094301235	0,27867654	0,099609877	3,883635	0,614523
Umf06.jpg	50	4r - 4c	223	139	1,770776583	8,009758662	0,184875	0,063292712	0,71911111	0,091617682	3,160717	0,554351
Umf06.jpg	60	8r - 4c	206	117	1,608085297	7,141517396	0,219807	0,059551066	0,25791246	0,112628507	2,435484	0,749697
Umf06.jpg	70	8r - 3c	172	103	1,772727669	5,489050109	0,184867	0,045546841	0,1962658	0,097599129	2,000275	0,629617
Umf06.jpg	80	8r - 3c	133	77	1,339534392	4,808279365	0,1808	0,074692063	0,17197884	0,13509418	1,627335	0,605304
Umf06.jpg	90	5r - 7c	138	81	1,255547325	4,594572016	0,267239	0,029436214	0,15460905	0,076534979	1,994181	0,765074
Umf06.jpg	100	8r - 4c	113	72	1,577711934	3,890226337	0,374724	0,032259259	0,15412757	0,123720165	1,506148	0,715576
Umf06.jpg	110	8r - 4c	101	50	1,37108642	3,417806584	0,18493	0,024325103	0,15258848	0,185329218	1,069918	0,649358
Umf06.jpg	120		107	49	2,009880658	3,582847737	0,32123	0,021868313	0,09797119	0,061419753	1,330021	0,624564
Umf06.jpg	130		83	41	1,098878879	2,452532533	0,213822	0,038774775	0,35659259	0,059691692	0,882767	0,630466
Umf06.jpg	140	8r - 3c	94	51	1,137921811	2,799790123	0,234177	0,041958848	0,16707819	0,101987654	1,059049	0,714403
Umf06.jpg	150	9r - 7c	95	54	0,98998999	3,046346346	0,190795	0,021317317	0,09370971	0,063627628	1,292845	0,632817
Umf07.jpg	10	5r - 3c	501	272	5,571156695	16,8405755	0,263031	0,122974359	0,35819943	0,167282051	6,341407	0,886473
Umf07.jpg	20	5r - 3c	250	136	3,10054321	8,588826038	0,205513	0,086038159	0,18797755	0,099586981	3,023479	0,573392
Umf07.jpg	30	5r - 4c	168	86	2,243431431	6,6998999	0,179732	0,049313313	0,13890691	0,124536537	1,741265	0,642787
Umf07.jpg	40	5r - 4c	123	63	2,038899471	4,401235979	0,181507	0,030848677	0,1256	0,077367196	1,534108	0,680381
Umf07.jpg	50	5r - 4c	105	51	1,779502222	3,666234074	0,158441	0,032918519	0,14619852	0,09453037	0,989304	1,19949
Umf07.jpg	60		83	41	1,972992593	2,765703704	0,187289	0,086903704	0,07877037	0,053244444	0,867985	0,511911
Umf07.jpg	70	5r - 3c	74	35	1,911827957	2,548033453	0,296272	0,020114695	0,42788053	0,057065711	0,993137	0,697405
Umf07.jpg	80		63	33	1,597648746	2,323956989	0,354366	0,013801673	0,06699164	0,057510155	0,759422	0,810065
Umf07.jpg	90		61	29	1,180114695	2,37604779	0,149864	0,020191159	0,05517324	0,038083632	1,076096	0,549066
Umf07.jpg	100		61	29	1,180114695	2,37604779	0,149864	0,020191159	0,05517324	0,038083632	1,076096	0,549066
Umf07.jpg	110		61	29	1,180114695	2,37604779	0,149864	0,020191159	0,05517324	0,038083632	1,076096	0,549066
Umf07.jpg	120		61	29	1,180114695	2,37604779	0,149864	0,020191159	0,05517324	0,038083632	1,076096	0,549066
Umf07.jpg	130		61	29	1,180114695	2,37604779	0,149864	0,020191159	0,05517324	0,038083632	1,076096	0,549066
Umf07.jpg	140		61	29	1,180114695	2,37604779	0,149864	0,020191159	0,05517324	0,038083632	1,076096	0,549066
Umf07.jpg	150		61	29	1,180114695	2,37604779	0,149864	0,020191159	0,05517324	0,038083632	1,076096	0,549066
Umf08.jpg	10	5r - 4c	383	161	5,50386141	12,48773238	0,26249	0,085863799	0,25095818	0,109471924	3,501744	0,667642
Umf08.jpg	20	5r - 4c	192	75	2,633033033	6,557161161	0,171203	0,038798799	0,10818819	0,060244244	1,603516	0,637121
Umf08.jpg	30	5r - 4c	129	57	3,019583584	4,470906907	0,188917	0,02243043	0,09390991	0,052188188	1,161145	0,587067
Umf08.jpg	40	5r - 4c	94	38	1,615425926	2,983892593	0,156619	0,017355556	0,08383704	0,047066667	0,781193	0,585544
Umf08.jpg	50		83	34	1,652744539	3,18865717	0,168885	0,014294397	0,05152137	0,067407407	0,814583	0,540551
Umf08.jpg	60		83	34	1,652744539	3,18865717	0,168885	0,014294397	0,05152137	0,067407407	0,814583	0,540551
Umf08.jpg	70	5r - 4c	57	25	2,032835979	1,869738977	0,175404	0,016186949	0,07182716	0,042772487	0,549556	0,837845
Umf08.jpg	80	5r - 4c	57	25	2,032835979	1,869738977	0,175404	0,016186949	0,07182716	0,042772487	0,549556	0,837845
Umf08.jpg	90		48	23	1,365209302	1,518146425	0,156755	0,012020672	0,05424634	0,034515073	0,5477	0,569668
Umf08.jpg	100		48	23	1,365209302	1,518146425	0,156755	0,012020672	0,05424634	0,034515073	0,5477	0,569668
Umf08.jpg	110		48	23	1,365209302	1,518146425	0,156755	0,012020672	0,05424634	0,034515073	0,5477	0,569668
Umf08.jpg	120		48	23	1,365209302	1,518146425	0,156755	0,012020672	0,05424634	0,034515073	0,5477	0,569668
Umf08.jpg	130		48	23	1,365209302	1,518146425	0,156755	0,012020672	0,05424634	0,034515073	0,5477	0,569668
Umf08.jpg	140		48	23	1,365209302	1,518146425	0,156755	0,012020672	0,05424634	0,034515073	0,5477	0,569668
Umf08.jpg	150		48	23	1,365209302	1,518146425	0,156755	0,012020672	0,05424634	0,034515073	0,5477	0,569668
Umf09.jpg	10	5r - 3c	1203	700	7,15047619	44,50391534	0,411767	0,434285714	0,92624339	0,32962963	14,32301	0,835661
Umf09.jpg	20	5r - 3c	601	339	4,17058642	20,36243827	0,685759	0,261919753	0,96038889	0,207641975	8,821926	1,182012
Umf09.jpg	30	5r - 3c	396	224	2,700863346	13,23907024	0,28002	0,116796935	0,35144828	0,171438059	5,415969	0,626641
Umf09.jpg	40	5r - 3c	304	159	2,090623094	10,89046623	0,257935	0,095821351	0,31509804	0,130169935	4,010763	0,629059
Umf09.jpg	50	5r - 3c	236	136	1,904035915	8,468740741	0,233607	0,065005612	0,24216835	0,111977553	2,996175	0,711187
Umf09.jpg	60	5r - 3c	192	103	1,469298246	6,622495127	0,221513	0,048265107	0,21441715	0,127953216	2,072043	0,657419
Umf09.jpg	70	4r - 2c	167	88	1,673866667	5,92365291	0,203831	0,063661376	0,41837037	0,101710053	2,054011	0,633689
Umf09.jpg	80	5r - 3c	147	74	1,493849953	4,828471035	0,192342	0,037648623	0,22119278	0,103479582	1,643335	0,629766
Umf09.jpg	90	5r - 3c	147	79	1,528609524	5,460427513	0,182976	0,033765079	0,31674497	0,099868783	1,738701	0,738383
Umf09.jpg	100	5r - 2c	122	64	1,191751852	4,0976	0,264359	0,033688889	0,14455926	0,085003704	1,507504	0,610363
Umf09.jpg	110	5r - 3c	106	61	1,16637037	3,458403704	0,179456	0,035159259	0,1661963	0,07607037	1,297819	0,653819
Umf09.jpg	120		97	43	1,127949318	3,05294347	0,236117	0,023017544	0,15318129	0,096561404	1,202394	0,529914
Umf09.jpg	130		83	46	1,160144349	2,704436847	0,175012	0,027836657	0,23707882	0,064471035	1,571449	0,618784
Umf09.jpg	140		82	42	1,006559387	2,740056194	0,156741	0,018871009	0,06367305	0,092301405	0,876679	0,631954
Umf09.jpg	150		82	42	1,006559387	2,740056194	0,156741	0,018871009	0,06367305	0,092301405	0,876679	0,631954
Umf10.jpg	10	9r - 3c	771	382	5,649699805	25,6725692	0,309692	0,198955166	0,71607018	0,217489279	11,13449	0,775135
Umf10.jpg	20	9r - 4c	383	195	3,823041975	13,79486914	0,261881	0,09574321	0,41077037	0,154360494	4,701358	0,651037
Umf10.jpg	30	9r - 4c	255	131	2,472592593	8,940647619	0,204305	0,070137566	0,19235556	0,095712169	2,769117	0,661393
Umf10.jpg	40		196	96	2,066474946	7,32854902	0,189364	0,043394336	0,14729847	0,11096732	2,21841	0,696824
Umf10.jpg	50		156	68	1,776980981	5,383855856	0,233165	0,02446046	0,08467267	0,064832833	1,380917	0,733217
Umf10.jpg	60		124	63	1,962506823	4,630019493	0,190402	0,024175439	0,22796881	0,064939571	1,45552	0,638378
Umf10.jpg	70	9r - 3c	109	56	1,42065717	4,556995252	0,181352	0,02137132	0,13494397	0,072934473	1,149318	0,675286
Umf10.jpg	80	6r - 0c	95	40	1,30768471	2,971187085	0,157094	0,01785755	0,07705603	0,066803419	0,918135	0,542663
Umf10.jpg	90		94	40	1,630534113	3,305754386	0,165099	0,018226121	0,0948694	0,081516569	0,95853	0,614651
Umf10.jpg	100		94	40	1,630534113	3,305754386	0,165099	0,018226121	0,0948694	0,081516569	0,95853	0,614651
Umf10.jpg	110		72	25	1,36127037	2,386244444	0,166704	0,016214815	0,06058519	0,052166667	0,545415	0,778715
Umf10.jpg	120		72	25	1,36127037	2,386244444	0,166704	0,016214815	0,06058519	0,052166667	0,545415	0,778715
Umf10.jpg	130		72	25	1,36127037	2,386244444	0,166704	0,016214815	0,06058519	0,052166667	0,545415	0,778715
Umf10.jpg	140		72	25	1,36127037	2,386244444	0,166704	0,016214815	0,06058519	0,052166667	0,545415	0,778715
Umf10.jpg	150		72	25	1,36127037	2,386244444	0,166704	0,016214815	0,06058519	0,052166667	0,545415	0,778715

Tabulka C.8: Pokračování výsledků testování na zařízení Nokia Lumia 620

Umf11.jpg	10	5r - 5c	770	351	5,68737037	24,69837037	0,501585	0,345948148	0,68017778	0,327496296	10,05481	0,94223
Umf11.jpg	20	9r - 4c	389	179	3,55083871	13,64524731	0,256779	0,086293907	0,25370609	0,122523297	4,525988	0,641027
Umf11.jpg	30	5r - 5c	257	115	2,574563412	9,064888889	0,214698	0,044754209	0,16940067	0,090361392	2,377899	0,658119
Umf11.jpg	40	5r - 4c	196	94	2,020098765	6,668439955	0,231367	0,066244669	0,26002245	0,086316498	2,197697	0,747677
Umf11.jpg	50		157	65	1,761526749	5,128028807	0,173975	0,037115226	0,09445679	0,074041152	1,327979	0,638173
Umf11.jpg	60		133	61	1,983183183	4,672816817	0,163788	0,025185185	0,0845045	0,051815816	1,548833	0,661994
Umf11.jpg	70		112	54	1,917961962	3,291063063	0,18199	0,021453453	0,07811011	0,058274274	1,169433	0,939315
Umf11.jpg	80		93	38	1,358233918	3,676105263	0,161368	0,017575049	0,0794191	0,040604288	0,891598	0,681622
Umf11.jpg	90		96	38	1,354849903	2,924471735	0,199064	0,011847953	0,04477193	0,035068226	1,09469	0,596558
Umf11.jpg	100		96	38	1,354849903	2,924471735	0,199064	0,011847953	0,04477193	0,035068226	1,09469	0,596558
Umf11.jpg	110		96	38	1,354849903	2,924471735	0,199064	0,011847953	0,04477193	0,035068226	1,09469	0,596558
Umf11.jpg	120		70	32	1,355809524	3,114256085	0,254421	0,035860317	0,0541291	0,035263492	0,775094	0,675886
Umf11.jpg	130		70	32	1,355809524	3,114256085	0,254421	0,035860317	0,0541291	0,035263492	0,775094	0,675886
Umf11.jpg	140		70	32	1,355809524	3,114256085	0,254421	0,035860317	0,0541291	0,035263492	0,775094	0,675886
Umf11.jpg	150		70	32	1,355809524	3,114256085	0,254421	0,035860317	0,0541291	0,035263492	0,775094	0,675886
Umf12.jpg	10	7r - 2c	947	521	5,737291005	36,16998942	0,370635	0,31421164	0,99280423	0,292740741	15,06234	0,69418
Umf12.jpg	20	7r - 3c	470	257	2,954021164	15,3727037	0,32236	0,134148148	0,46190476	0,20078836	5,947439	0,672095
Umf12.jpg	30	7r - 3c	324	165	2,343416667	10,92795833	0,306394	0,066259259	0,304625	0,123694444	3,693338	0,657458
Umf12.jpg	40	7r - 3c	238	122	2,061708061	7,932962963	0,252288	0,065956427	0,2050719	0,090884532	2,724305	0,971046
Umf12.jpg	50	7r - 3c	198	104	2,34535873	6,254480423	0,230942	0,040444444	0,18151534	0,083259259	2,091907	0,867865
Umf12.jpg	60	7r - 3c	165	79	1,789733734	5,297013013	0,203784	0,024660661	0,35536336	0,079831832	1,789542	0,92784
Umf12.jpg	70	8r - 2c	143	70	1,535111111	4,843041152	0,229539	0,03654321	0,16641975	0,102230453	1,744877	0,650185
Umf12.jpg	80	7r - 3c	117	62	2,101585586	3,845069069	0,249506	0,034750751	0,1036997	0,074542543	1,255067	0,722322
Umf12.jpg	90	6r - 6c	112	53	1,844386831	3,568489712	0,181128	0,017781893	0,07054733	0,061185185	1,095617	0,786309
Umf12.jpg	100	7r - 3c	100	51	1,235825926	3,09087037	0,187633	0,02082963	0,11068519	0,065185185	1,302322	0,696189
Umf12.jpg	110	8r - 2c	99	46	1,24382963	3,43507037	0,194748	0,017959259	0,08282222	0,067140741	1,026852	0,68177
Umf12.jpg	120		95	41	1,135760684	2,725565052	0,208418	0,016581197	0,04951567	0,044528015	0,958929	0,528186
Umf12.jpg	130		95	41	1,135760684	2,725565052	0,208418	0,016581197	0,04951567	0,044528015	0,958929	0,528186
Umf12.jpg	140	8r - 2c	74	32	1,021723577	2,351082204	0,175707	0,017691057	0,09873532	0,071985547	1,100311	0,631151
Umf12.jpg	150	7r - 1c	78	36	1,084426378	2,261315266	0,173868	0,016588979	0,09071364	0,060086721	0,804014	0,721088
Umf13.jpg	10	6r - 3c	1111	313	6,105287749	34,88575499	0,358883	0,178176638	0,53187464	0,261766382	8,33388	1,038678
Umf13.jpg	20	6r - 4c	569	145	3,411356125	17,58708832	0,221151	0,081356125	0,21826781	0,114307692	3,091829	0,635373
Umf13.jpg	30	7r - 3c	364	103	2,519832099	10,42952593	0,213807	0,049249383	0,19205432	0,128716049	2,09118	0,699131
Umf13.jpg	40	-2r - 3c	283	67	2,122567901	8,310908642	0,203136	0,031441975	0,12105679	0,058992593	1,40282	0,847763
Umf13.jpg	50	5r - 3c	226	56	1,744215488	6,611618406	0,210384	0,021948373	0,09210325	0,059196409	1,416911	0,59068
Umf13.jpg	60		183	47	1,698704762	5,644486772	0,20008	0,025037037	0,09209735	0,083589418	1,016957	0,53873
Umf13.jpg	70		164	41	1,387695238	4,942247619	0,195166	0,021828571	0,07826032	0,029130159	0,869477	0,569731
Umf13.jpg	80		164	41	1,387695238	4,942247619	0,195166	0,021828571	0,07826032	0,029130159	0,869477	0,569731
Umf13.jpg	90		133	34	1,431492063	3,979128042	0,165037	0,026514286	0,06674709	0,067686772	1,128876	0,582404
Umf13.jpg	100		124	23	1,331987654	3,640761317	0,321263	0,018082305	0,04401235	0,045954733	0,695354	0,557305
Umf13.jpg	110		113	31	1,675366539	4,101445722	0,219576	0,018993614	0,08755556	0,046232439	0,824046	0,583418
Umf13.jpg	120		113	31	1,675366539	4,101445722	0,219576	0,018993614	0,08755556	0,046232439	0,824046	0,583418
Umf13.jpg	130		113	31	1,675366539	4,101445722	0,219576	0,018993614	0,08755556	0,046232439	0,824046	0,583418
Umf13.jpg	140		113	31	1,675366539	4,101445722	0,219576	0,018993614	0,08755556	0,046232439	0,824046	0,583418
Umf13.jpg	150		113	31	1,675366539	4,101445722	0,219576	0,018993614	0,08755556	0,046232439	0,824046	0,583418
Umf14.jpg	10	4r - 3c	793	332	5,310361392	23,06698092	0,250994	0,149912458	0,34644669	0,163631874	7,97073	0,682213
Umf14.jpg	20	5r - 3c	399	171	2,77299839	11,44251208	0,223469	0,073487923	0,20087923	0,124077295	3,463272	0,607504
Umf14.jpg	30	5r - 3c	263	106	2,254904866	7,828174292	0,15955	0,042678286	0,13882934	0,077275236	1,790812	0,560871
Umf14.jpg	40	14r - 6c	203	99	1,740219668	6,334411239	0,178064	0,035417625	0,13678161	0,075016603	1,711716	0,670204
Umf14.jpg	50	6r - 7c	165	67	1,853475891	4,958627533	0,154342	0,030350804	0,11373585	0,083698113	1,199019	0,597191
Umf14.jpg	60	5r - 4c	138	60	1,554713698	4,048159906	0,188127	0,023539095	0,12783304	0,063466196	1,136508	0,563104
Umf14.jpg	70	9r - 7c	115	43	1,285115741	3,278087963	0,168889	0,016289352	0,08076157	0,051425926	0,791856	0,64219
Umf14.jpg	80		99	39	1,344941799	3,558768959	0,155915	0,014419753	0,0665291	0,053516755	0,982321	0,587093
Umf14.jpg	90		96	33	1,944129155	2,81508452	0,175594	0,015981007	0,06149288	0,043536562	0,949033	0,605899
Umf14.jpg	100	15r - 9c	87	36	1,539297246	3,098423552	0,173216	0,020307692	0,09061728	0,075019943	0,819069	0,7637
Umf14.jpg	110		75	30	1,320074074	2,280074074	0,164963	0,015121693	0,06031746	0,041985891	0,671115	0,536123
Umf14.jpg	120		75	30	1,320074074	2,280074074	0,164963	0,015121693	0,06031746	0,041985891	0,671115	0,536123
Umf14.jpg	130		75	30	1,320074074	2,280074074	0,164963	0,015121693	0,06031746	0,041985891	0,671115	0,536123
Umf14.jpg	140		75	30	1,320074074	2,280074074	0,164963	0,015121693	0,06031746	0,041985891	0,671115	0,536123
Umf14.jpg	150		75	30	1,320074074	2,280074074	0,164963	0,015121693	0,06031746	0,041985891	0,671115	0,536123
Umf15.jpg	10	3r - 3c	2020	889	7,298008889	66,85734222	0,53032	0,580885926	1,09185778	0,355872593	20,57485	0,576856
Umf15.jpg	20	7r - 4c	1007	457	3,741507625	33,22063617	0,503939	0,3023878	0,7124793	0,350440087	12,77378	0,798806
Umf15.jpg	30	3r - 4c	661	277	2,509703704	21,03738462	0,318684	0,151071225	0,53270085	0,220319088	6,466479	0,699271
Umf15.jpg	40	4r - 3c	505	231	2,019703704	15,74591495	0,24524	0,12260631	0,67948971	0,167917695	5,342091	0,605311
Umf15.jpg	50	4r - 3c	416	185	1,855985185	12,75437531	0,249215	0,089975309	0,2873679	0,145802469	4,299077	0,977309
Umf15.jpg	60	4r - 3c	320	145	1,906437276	9,853252091	0,237204	0,081753883	0,2873644	0,11339546	3,007632	0,600879
Umf15.jpg	70	4r - 3c	289	124	1,552123457	9,296057613	0,19486	0,099164609	0,24370782	0,105798354	2,448852	0,564951
Umf15.jpg	80		249	103	1,463921569	7,730143791	0,17546	0,048618736	0,16388671	0,09871024	2,131307	0,624828
Umf15.jpg	90		209	82	1,908093122	6,705663492	0,17945	0,053299471	0,20145185	0,1096	1,895272	0,654908
Umf15.jpg	100	9r - 5c	210	95	1,289485485	7,533761762	0,226899	0,044492492	0,27247648	0,090794795	1,961401	0,655868
Umf15.jpg	110		184	69	1,397230769	5,791468186	0,174952	0,032630579	0,14423932	0,095855651	1,433763	0,574838
Umf15.jpg	120	3r - 8c	161	73	1,248622982	4,853154796	0,200368	0,026598291	0,11692688	0,063350427	1,714283	0,580813
Umf15.jpg	130	2r - 9c	150	73	1,403832858	5,191130104	0,181314	0,031274454	0,27607977	0,107665717	1,491821	0,595844
Umf15.jpg	140	12r - 5c	150	67	1,170760614	4,953365854	0,17557	0,030825655	0,1764589	0,075226739	1,41221	0,705677
Umf15.jpg	150		136	65	1,242200542	4,204379404	0,272759	0,027457995	0,09087624	0,057763324	1,519089	0,563537

Tabulka C.9: Pokračování výsledků testování na zařízení Nokia Lumia 620

Umf16.jpg	10	6r - 3c	2472	847	10,00341595	79,66051567	0,49363	0,59845584	0,92184615	0,418361823	19,78766	0,574681
Umf16.jpg	20	6r - 3c	1214	413	4,736237037	38,98961975	0,390795	0,256987654	0,71087407	0,236632099	11,15375	0,898815
Umf16.jpg	30	6r - 2c	818	264	3,468972625	24,20780032	0,339266	0,152676329	0,72769726	0,187375201	8,164599	0,770106
Umf16.jpg	40	6r - 3c	619	206	2,769805213	18,3558299	0,262667	0,102002743	0,31008505	0,168910837	5,004044	0,748626
Umf16.jpg	50	6r - 1c	511	164	2,600510856	16,17690932	0,215162	0,076398467	0,27594891	0,416439336	3,905829	0,696322
Umf16.jpg	60	6r - 3c	408	129	1,776171296	12,14353241	0,26112	0,07762037	0,27823148	0,171486111	2,806407	0,712625
Umf16.jpg	70	7r - 3c	351	110	2,047054466	10,83432244	0,214471	0,045712418	0,20312854	0,107795207	2,520911	0,723198
Umf16.jpg	80	0r - 10c	305	98	1,327980676	8,991826087	0,287285	0,031446055	0,16161031	0,080431562	1,948283	0,646969
Umf16.jpg	90	7r - 3c	280	81	1,823580952	8,641506878	0,232123	0,034721693	0,21686349	0,100520635	1,792322	0,728063
Umf16.jpg	100		257	75	1,453020576	7,49655144	0,190872	0,047296296	0,12430864	0,089016461	1,517988	0,650547
Umf16.jpg	110	7r - 2c	246	69	1,268246914	7,157662551	0,404111	0,046773663	0,22616461	0,084897119	1,968486	0,678317
Umf16.jpg	120	7r - 4c	219	69	1,560724725	6,399143143	0,204204	0,035287287	0,18223023	0,126394394	1,673077	0,760541
Umf16.jpg	130	7r - 5c	182	60	1,103190883	5,2828566	0,203749	0,023992403	0,11757265	0,068710351	1,226777	0,655104
Umf16.jpg	140		187	54	1,165883041	5,877415205	0,16554	0,017933723	0,08796101	0,055547758	1,203021	0,600347
Umf16.jpg	150	7r - 6c	198	59	1,225837838	5,203367367	0,188589	0,022786787	0,09804605	0,049245245	1,291427	0,692617
Umf17.jpg	10	4r - 6c	1932	777	8,365022222	65,06839704	0,49053	0,476245926	0,98304	0,310085926	17,942	0,735828
Umf17.jpg	20	8r - 5c	980	386	3,755160494	31,2154321	0,42656	0,276263374	0,71837037	0,329069959	10,77816	1,264963
Umf17.jpg	30	4r - 5c	627	257	2,837582222	20,01103407	0,300059	0,14802963	0,43077926	0,216965926	6,149487	0,835176
Umf17.jpg	40	4r - 6c	477	181	2,294696296	14,7861037	0,38842	0,10222716	0,39368395	0,222883951	4,181901	0,837368
Umf17.jpg	50	4r - 6c	385	164	2,060828704	11,68953241	0,247773	0,115597222	0,27343519	0,171138889	3,596264	0,866648
Umf17.jpg	60	4r - 5c	315	119	1,582684096	9,403407407	0,313647	0,070418301	0,24902397	0,110562092	2,922436	0,767142
Umf17.jpg	70	4r - 7c	297	105	1,845102881	8,363868313	0,224868	0,075884774	0,40189712	0,128506173	2,148165	0,813597
Umf17.jpg	80	4r - 6c	247	94	1,353557672	7,376389418	0,213494	0,075729101	0,22072804	0,103754497	2,029676	0,828622
Umf17.jpg	90	5r - 6c	231	84	1,74768254	6,810035979	0,223839	0,04650582	0,1936254	0,195140741	1,843039	0,635856
Umf17.jpg	100		202	81	1,928796797	6,207771772	0,226298	0,042830831	0,18296296	0,078282282	1,754583	0,574783
Umf17.jpg	110	7r - 5c	190	74	1,218390947	5,95237037	0,264547	0,02944856	0,14320988	0,07881893	1,949091	0,787016
Umf17.jpg	120	9r - 7c	150	50	1,32002963	4,193285185	0,20057	0,026944444	0,08601852	0,069025926	1,09887	0,672667
Umf17.jpg	130		163	56	1,148094967	4,172957265	0,359867	0,021340931	0,10248053	0,058723647	1,494644	0,887081
Umf17.jpg	140		157	48	1,115526649	3,794883469	0,199599	0,030301716	0,08651491	0,067642276	1,052177	0,551357
Umf17.jpg	150	5r - 11c	146	59	1,222153846	4,032121557	0,1855	0,024888889	0,13441216	0,082275404	1,376038	0,625546
Umf18.jpg	10		1659	414	10,06594239	49,17514403	0,558617	0,475522634	0,58153086	0,251555556	11,59618	0,769794
Umf18.jpg	20	4r - 3c	840	213	3,604897603	22,33071895	0,39607	0,129917211	0,41037037	0,344252723	5,121281	1,644218
Umf18.jpg	30	6r - 3c	561	137	2,564716846	15,57933094	0,254346	0,069189964	0,27562724	0,145495818	2,966519	0,850323
Umf18.jpg	40	7r - 4c	432	104	1,963526882	12,01644444	0,255924	0,066547192	0,1852951	0,109400239	2,232239	1,0187
Umf18.jpg	50	6r - 5c	331	85	1,962867102	8,622901961	0,220784	0,046039216	0,16850545	0,081250545	1,844013	0,705072
Umf18.jpg	60		282	64	2,174334334	7,674726727	0,204352	0,027575576	0,08053654	0,057165165	1,315928	0,63793
Umf18.jpg	70		237	63	1,774998999	6,561165165	0,207996	0,050782783	0,10615015	0,068784785	1,375011	0,800937
Umf18.jpg	80	13r - 6c	207	50	1,331419419	5,402214214	0,406519	0,026718719	0,09755355	0,079655656	1,186647	0,605073
Umf18.jpg	90		199	41	1,445801802	4,872256256	0,266971	0,049189189	0,11667668	0,057225225	0,911772	0,576905
Umf18.jpg	100	9r - 5c	165	45	1,597082621	4,566807217	0,225311	0,028467236	0,11185185	0,059920228	1,07491	0,601915
Umf18.jpg	110		157	35	1,3908577	4,785122807	0,184055	0,013727096	0,05450682	0,045793372	0,759509	0,591173
Umf18.jpg	120		146	30	1,378253411	3,392697856	0,179466	0,012506823	0,04094737	0,049575049	0,723828	0,778008
Umf18.jpg	130		124	27	1,060744354	3,248957543	0,181312	0,016238482	0,05417886	0,050178862	0,63927	0,577427
Umf18.jpg	140		108	26	1,171157895	2,666253411	0,18391	0,015302144	0,06660819	0,039224172	0,593641	0,638487
Umf18.jpg	150		115	29	1,093485185	3,10452963	0,211837	0,022959259	0,05458519	0,042774074	0,771878	0,631796
Umf19.jpg	10	7r - 0c	763	423	6,489917211	28,12084532	0,40149	0,370928105	0,56577778	0,254257081	11,0784	0,619242
Umf19.jpg	20	7r - 1c	382	208	3,06497284	13,2302963	0,256479	0,11177284	0,33179753	0,171797531	4,819906	0,6336
Umf19.jpg	30	7r - 1c	254	136	2,912126362	9,094749455	0,204758	0,077546841	0,19585185	0,096261438	3,309185	0,692052
Umf19.jpg	40	7r - 1c	199	106	1,82072428	6,977012346	0,178881	0,03572428	0,15075309	0,125415638	2,059412	0,668235
Umf19.jpg	50	7r - 0c	162	87	1,552760761	5,334790791	0,202503	0,061597598	0,1430951	0,085581582	2,154126	0,797618
Umf19.jpg	60	7r - 1c	131	67	1,577750487	4,593438596	0,163891	0,041302144	0,11690838	0,077072125	1,372565	0,602651
Umf19.jpg	70	7r - 1c	106	58	1,448668566	3,783563153	0,178792	0,026932574	0,10735802	0,103365622	1,230659	0,616961
Umf19.jpg	80		91	49	1,272227642	3,19130262	0,177485	0,021217706	0,09169286	0,097669377	1,118042	0,563595
Umf19.jpg	90		92	45	1,265116809	3,326575499	0,170465	0,017458689	0,11325736	0,092345679	1,192118	0,498511
Umf19.jpg	100		80	43	1,237892593	2,703	0,1968	0,019077778	0,10323704	0,0493	0,870022	0,674226
Umf19.jpg	110		71	38	1,252462963	2,386437037	0,219515	0,018892593	0,07515556	0,083118519	0,901685	0,650667
Umf19.jpg	120		69	32	1,354157182	2,794247516	0,170341	0,014218609	0,11501355	0,04964047	0,788925	0,536448
Umf19.jpg	130		60	27	1,074218519	1,831514815	0,151207	0,013911111	0,04974444	0,027955556	0,732867	0,833433
Umf19.jpg	140		52	29	1,295733333	1,899088889	0,157033	0,014866667	0,06416667	0,071955556	0,651456	0,605089
Umf19.jpg	150		52	29	1,295733333	1,899088889	0,157033	0,014866667	0,06416667	0,071955556	0,651456	0,605089
Umf20.jpg	10	4r - 7c	1954	876	7,259837328	65,80840087	0,48583	0,602730574	1,11742338	0,449475672	20,00807	0,696645
Umf20.jpg	20	4r - 7c	975	425	3,919555556	33,33031808	0,419913	0,305054466	0,99526797	0,323538126	10,44807	0,696436
Umf20.jpg	30	7r - 3c	640	281	2,691608889	19,65795556	0,320533	0,135075556	0,45507556	0,175774815	6,821683	0,622933
Umf20.jpg	40	4r - 7c	475	207	2,826714074	15,02087704	0,238756	0,119152593	0,31928296	0,195134815	4,983923	0,587093
Umf20.jpg	50	3r - 6c	404	171	2,529247312	13,88533811	0,276468	0,119125448	0,3345902	0,181959379	4,879718	0,752177
Umf20.jpg	60	7r - 3c	315	139	1,780645161	10,3471828	0,204516	0,064482676	0,2887264	0,13148865	3,070141	0,635302
Umf20.jpg	70	7r - 3c	272	113	1,514325477	8,416929293	0,183376	0,056264871	0,2684624	0,155488215	2,261033	0,603012
Umf20.jpg	80	7r - 4c	219	89	1,589245245	7,338026026	0,20165	0,049573574	0,21754154	0,254466466	1,780745	0,572084
Umf20.jpg	90	6r - 8c	230	94	1,860711934	7,409283951	0,16358	0,038641975	0,1602716	0,167436214	2,080979	0,717658
Umf20.jpg	100		207	87	1,692834308	6,799758285	0,181965	0,056011696	0,14247563	0,122475634	2,209392	0,598168
Umf20.jpg	110		197	76	1,454183236	5,772557505	0,168639	0,040405458	0,17269006	0,103610136	1,515072	0,611684
Umf20.jpg	120		158	67	1,232244444	5,484659259	0,242674	0,021855556	0,08421111	0,054459259	1,325933	0,551948
Umf20.jpg	130	4r - 5c	156	71	1,333407407	4,431504873	0,18285	0,029559454	0,12814035	0,071808967	1,552327	0,665485
Umf20.jpg	140		140	58	1,054925926	4,494848148	0,165441	0,032044444	0,12450741	0,076148148	1,340393	0,558911
Umf20.jpg	150		137	58	1,224118519	4,157977778	0,191111	0,028722222	0,08986296	0,053814815	1,201933	0,714478