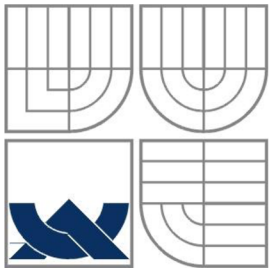
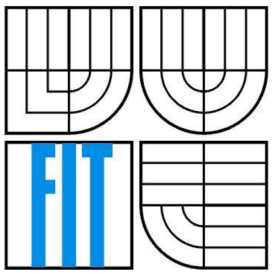


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ OBJEKTŮ V OBRAZE NA ZÁKLADĚ HRAN

OBJECT DETECTION BASED ON EDGES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Jaroslav Caha
AUTHOR

VEDOUCÍ PRÁCE — Ing. Michal Španěl
SUPERVISOR

BRNO 2010

Abstrakt

Prezentovaná metoda detekuje ve vstupním obraze dveře na základě jejich hran. Je důležité dveře odlišit od podobných objektů jako okno nebo vzor na podlaze. Proto je snímek rozdělen na části (podlaha, stěna, strop) a potenciální poloha dveří je tak lépe vymezena. Předpokládá se využití v robotech, kteří se pohybují uvnitř budov.

Abstract

This work presents a door detection method in images for mobile robot navigation. The method is able to detect doors in an input picture on the basis of found image edges. It is important to distinguish the door from similar objects like windows, paintings, or floor patterns. Therefore, the picture is divided into more parts (a floor, a wall, a ceiling) so that the potential placement of the door can be better drawn.

Klíčová slova

Detekce dveří, rozpoznávání objektů, hrany, Canny, Hough, Vanishing point, RANSAC, interiér

Keywords

Door detection, object recognition, edges, Canny, Hough, Vanishing point, RANSAC, indoor scenes

Citace

Caha Jaroslav: Rozpoznávání objektů v obraze na základě hran, bakalářská práce, Brno, FIT VUT v Brně, 2010

ROZPOZNÁVÁNÍ OBJEKTŮ V OBRAZE NA ZÁKLADĚ HRAN

Prohlášení

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

Jaroslav Caha
13. května 2010

Poděkování

Děkuji Ing. Michalu Španělovi za inspirativní vedení projektu a přínosné rady technického i organizačního rázu.

OBSAH

1 ÚVOD

Algoritmy pro zpracování obrazu jsou využívány v mnoha aplikacích, které slouží pro pochopení a zpracování reality počítačem. Setkáme se s nimi jak v průmyslu, tak v běžném životě. Ve velké míře například ve zdravotnictví nebo v zábavní elektronice.

Moje práce se zabývá zpracováním obrazu v robotech. Konkrétně rozpoznáváním dveří. Využití je plánováno u robotů, kteří se pohybují uvnitř budov. Pro robota je nutné získat pomocí různých senzorů co nejvíce informací o místě, ve kterém pracuje. Například kde přesně je podlaha, stěny nebo různé průchody či překážky.

Cílem projektu je navrhnout a úspěšně otestovat systém, který poskytne robotovi informaci o tom, kde se nachází zavřené dveře. Díky tomu bude robot schopen dveře otevřít, případně pouze zmapovat jejich polohu pro pozdější využití. Obraz z robotovy kamery bude zpracován v několika krocích. Nejdříve bude Cannyho hranovým detektorem vytvořena hranová reprezentace vstupního obrazu. V ní budou Houghovou transformací detekovány rovné čáry. Vzájemná poloha těchto čar umožní určit pomocí algoritmu RANSAC Vanishing point a tím definovat, kde se v obraze nachází zeď. V posledním kroku budou zpracovány rovné čáry a označeny dveře.

Práce začíná nastíněním známých přístupů k detekci dveří ve druhé kapitole. Třetí kapitola popisuje kompletní návrh systému pro detekci dveří tak, jak byl skutečně vytvořen. Obsahuje také popis důležitých metod, které bylo nutné pro úspěšné vyřešení problému vyvinout. Ve čtvrté kapitole jsou rozebrány detaily implementace programu. Také se seznámíme s podrobným popisem všech parametrů, které slouží k nastavení jednotlivých metod pro detekci dveří. Poslední pátá kapitola shrnuje převážně obrazovou formou dosažené výsledky. Závěr nabízí případné možnosti vylepšení stávajícího systému a popisuje jeho přínos.

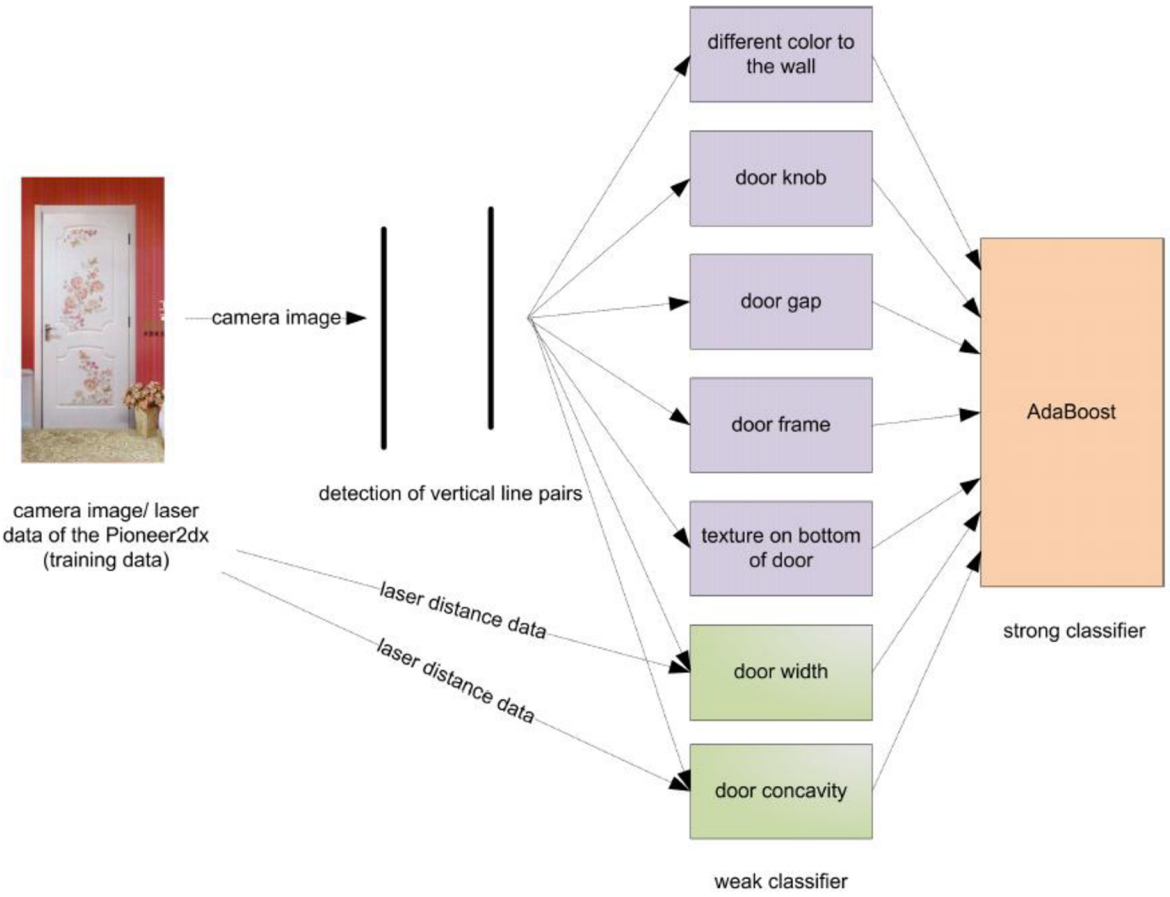
2 EXISTUJÍCÍ PŘÍSTUPY K DETEKCI DVEŘÍ

Při studování detekce dveří jsem přečetl několik článků, které se danou problematikou zabývají. Ve své práci jsem se jimi inspiroval.

Článek *Visual Detection of Lintel-Occluded Doors from a Single Image* [2] popisuje detekci dveří na základě výsledků šesti klasifikátorů, které se zaměřují na následující věci:

- Hrany dveří
- Mezera pode dveřmi
- Vanishing point
- Ocelová deska na spodní straně dveří (Kick plate)
- Textura dveří
- Barva dveří

Metodou, kterou v článku nazývají AdaBoost, přiřadí jednotlivým klasifikátorům určité váhy, které pomohou posoudit, jakou měrou se dílčí výsledky budou podílet na konečném verdiktu. Asi nejzajímavější částí je Vanishing point. Jedná se o průsečík hran podlahy a horních a spodních hran dveří. Jeho poloha je vypočítána jako střední hodnota průsečíků nesvislých čar v obraze.



Obr. 2.1: Detekce dveří v článku [3]

Velmi podobný způsob detekce je popsán také v článku *Real-time Door Detection based on AdaBoost learning algorithm* [3]. V článku používají více klasifikátorů a navíc i laserové měřidlo. Schéma detekce je na Obr. 2.1. Obrázek jsem převzal z článku. Vidíme na něm seznam sedmi slabých klasifikátorů, z nichž je algoritmem AdaBoost vytvořen jeden silný klasifikátor, který říká, zda se jedná o dveře či nikoliv.

Ve své práci jsem využil poznatky z výše citovaných dokumentů. Vycházím, podobně jako autoři článku *Real-time Door Detection based on AdaBoost learning algorithm* [3], ze dvou svislých čar. Dále v jejich blízkosti hledám čáry nesvislé, které by mohly představovat horní a spodní hranu dveří. Provádím výpočet Vanishing pointu a lokalizaci dveří pomocí jasového histogramu. Využívám tedy tří klasifikátorů, podle nichž je určen konečný výsledek.

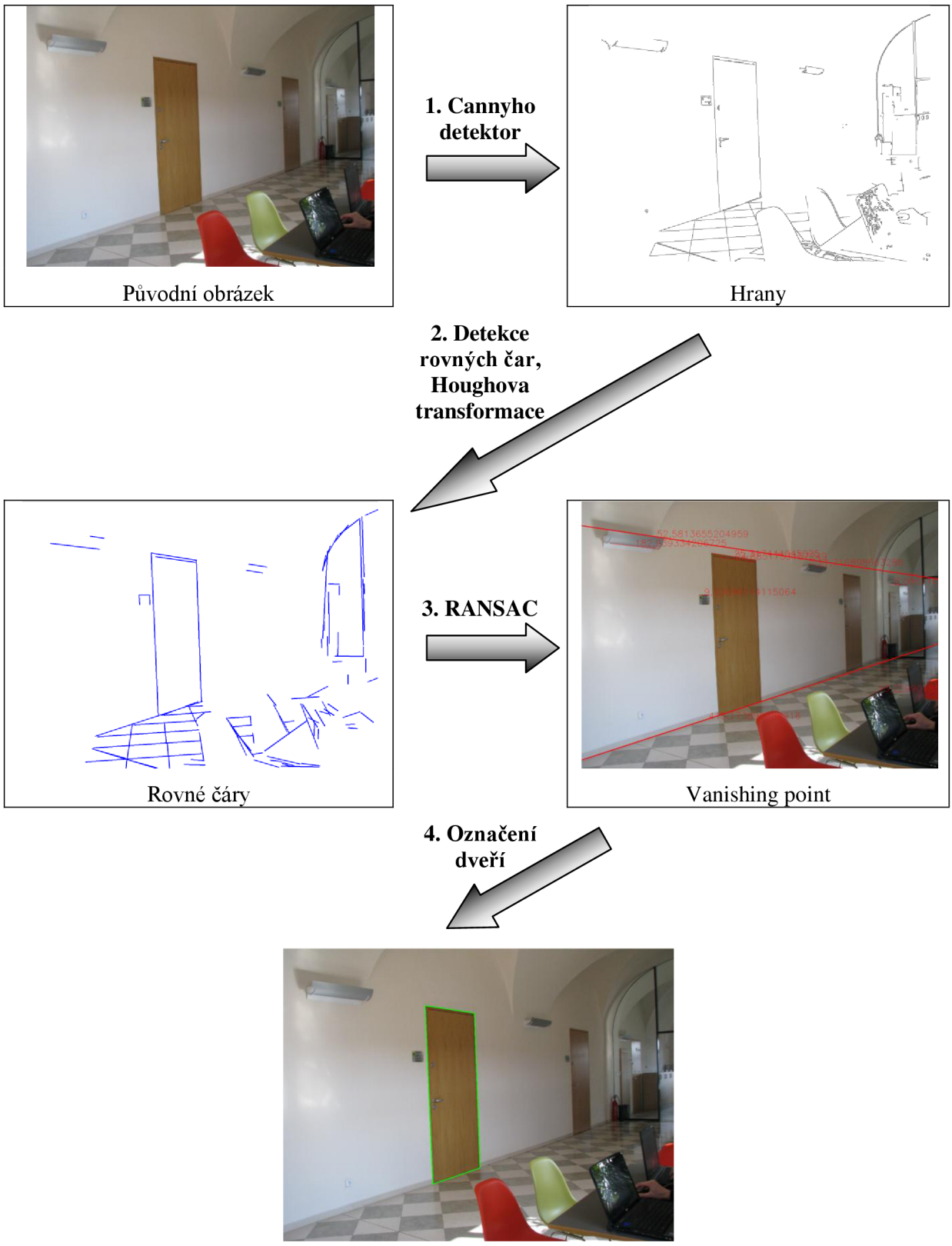
3 NÁVRH

Detekci dveří je možné rozdělit na několik funkčních celků, jak bylo naznačeno výše. Každý z nich produkuje důležité informace, které budou využity k výpočtu celkového výsledku. V této kapitole bude podrobně popsána funkce jednotlivých bloků. Na

Obr. 3.1 je zobrazena návaznost jednotlivých bloků.

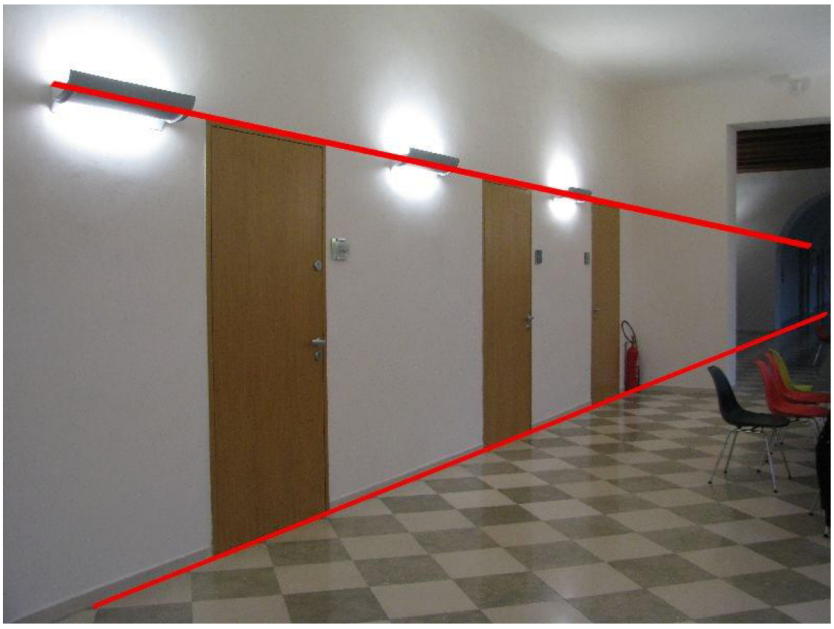
Nejdříve je nutné získat zjednodušený pohled na obrázek, tedy jeho hranovou reprezentaci. Při použití hranového detektoru získáme sice menší, ale stále příliš velké množství dat. Každý detail, jako listy květin, vzor na stěně nebo papírek na podlaze bude na výstupu zobrazen.

V případě hledání dveří nejsou takové maličkosti důležité. V obraze působí rušivě a mohou negativně ovlivnit celkový výsledek. Proto je vhodné krátké čáry vůbec nebrat v úvahu, čímž se zmenší objem dat a zjednoduší se následné zpracování. V dalším kroku bude potřeba vybrat ze všech hran pouze ty, které by potenciálně mohly označovat dveře. Jedná se výhradně o rovné čáry. To znamená, že ve výstupu hranového detektoru bude nutné najít úsečky a zapsat jejich polohu. Nejlépe jako souřadnice krajních bodů, ze kterých lze snadno získat rovnice těchto přímek a dále s nimi počítat.



Obr. 3.1: Schematické znázornění detekce dveří.

Nyní jsme v situaci, kdy místo vstupního obrázku máme k dispozici seznam všech rovných čar, jejichž délka byla omezena určitou minimální hodnotou. S těmito daty již je možné provádět různé výpočty. Budeme tedy zjišťovat, kde se na obrázku nachází stěna. Tím bude přesně vymezeno, kde se dveře mohou nacházet. Hledaný prostor nám pomůže definovat takzvaný Vanishing point. Jedná se o pomyslný průsečík, do kterého směřují všechny vodorovné hrany na chodbě. Především hrana mezi podlahou a stěnou, horní a dolní hrany dveří a případně hrana mezi stěnou a stropem. Na Obr. 3.2 je znázorněno, jak by měly vypadat stěžejní čáry, které určí Vanishing point. Čáry neprochází pouze dveřmi, ale také osvětlením, které v tomto případě kopíruje horní hranu dveří.



Obr. 3.2: Hrany definující Vanishing point.

Ve většině případů je možné na obrázcích nalézt mnoho různých předmětů, které jsou orientované tak, že směřují do Vanishing pointu. Toho je potřeba využít a hledání navrhnout tak, aby do výpočtu bylo zahrnuto pokud možno co nejvíce relevantních čar. Naproti tomu mohou výsledek negativně ovlivnit předměty, které Vanishing point neurčují, ale je jich ve scéně velké množství. V uvedeném obrázku se jedná například o dlažbu, která bude generovat velké množství čar, které jistě budou směřovat do jednoho bodu, ale ne do Vanishing pointu.

Až se podaří správně určit Vanishing point, přijde řada na poslední část, tedy hledání dveří. Je zřejmé, že musí být na stěně. Díky tomu se prohledávaná oblast obrázku výrazně zmenší. Detekce bude kombinovat informace o vzájemné poloze svislých čar a informace o změně barvy v okolí těchto čar. Pokud budou dvě čáry od sebe vhodně daleko a zároveň bude u obou z nich nalezena nápadná změna barvy, je velmi pravděpodobné, že se jedná o dveře. Pro potvrzení budou ještě hledány spodní a horní hrany dveří, které by měly směřovat do Vanishing pointu.

3.1 Detekce hran

V první fázi převedeme barvy v obrázku na stupně šedi. Potom pomocí detektoru hran získáme hranovou reprezentaci obrázku. Používám Cannyho detektor hran. Je obecně považován za nejlepší hranový detektor. Nyní je potřeba z obrázku vybrat ty hrany, které jsou rovné, tedy pro další

výpočty použitelné. V obrázku hledáme dveře, které mají dvě svislé hrany a dvě šikmé, které směřují do Vanishing pointu. Potřebujeme však znát souřadnice krajních bodů těchto čar, nestačí obrázek, který je výstupem Cannyho detektoru.

Algoritmy pro detekci rovných čar

K vyhledání různých geometrických útvarů v množině bodů lze použít Houghovu transformaci. Na základě rovnice hledaného útvaru je prohledáván vstupní obraz a výsledkem jsou souřadnice hraničních bodů všech nalezených útvarů. Při testování této metody jsem však dospěl k názoru, že se pro použití v mé práci nehodí. Její výsledky nesplňovaly očekávání. Výsledné čáry neodpovídaly svým předlohám a nemohly být dále použity pro detekci dveří. Podrobněji se k tomuto problému vyjádřím v kapitole Implementace.

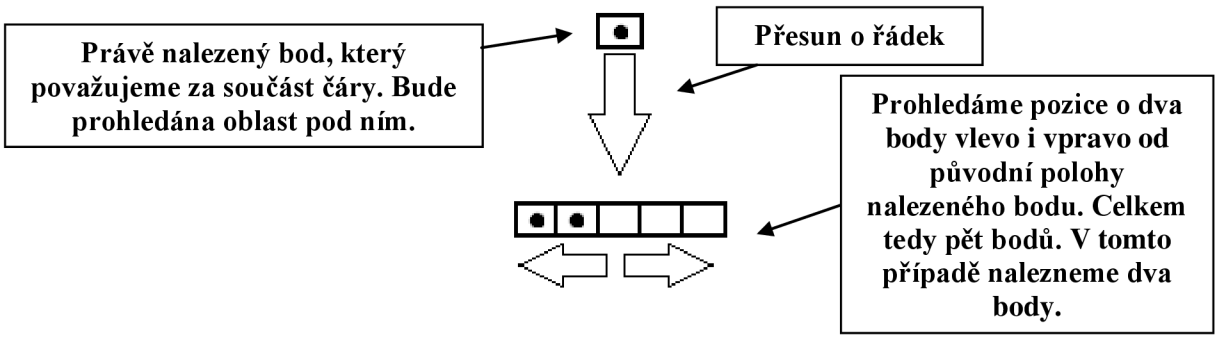
Rozhodl jsem se vytvořit vlastní algoritmus pro detekci rovných čar, který bude poskytovat výsledky vhodné pro další zpracování. Základní požadavek je, aby byly čáry detekovány co nejvěrněji. Tedy dlouhá čára by měla být detekována jako jedna čára, ne jako více čar, které budou od sebe mírně odsazeny. Také je třeba, aby nalezené čáry co nejlépe kopírovaly svoji předlohu. Úplné přesnosti však dosáhnout nelze, protože vstupní obraz trpí různými vadami. Například takzvaná *„soudkovitost“*. Je způsobena nedokonalostí objektivu fotoaparátu, který zobrazuje rovné svislé čáry mírně zahnuté. Je třeba mít tuto vlastnost všech fotografií na paměti a podle toho k detekci přistupovat. Detekovaná čára bude tedy vždy pouze určitou aproximací původní čáry.

Detekci čar jsem rozdělil na dva kroky. Nejdříve budu zpracovávat čáry svislé, potom ty ostatní. Nebude se jednat o čáry vodorovné, protože snímaný prostor je ovlivněn perspektivou, čáry se tedy sbíhají a jsou šikmé. Čáry jiné než svislé budu pro jednoduchost označovat jako *„nesvislé“*.

Detekce svislých čar

Víme, že hledáme rovnou čáru, která prochází obrázkem shora dolů. Čára může být kostrbatá, zahnutá a mohou ji protínat jiné čáry. Pokud se pokusíme ohraničit oblast, kde se taková čára může vyskytovat, dostaneme útvar podobný obdélníku. Obdélník může být i prohnutý nebo mírně zešikmený, vzhledem k výše uvedeným vlastnostem fotografií. Obrázek budeme procházet po řádcích, a když najdeme hranový bod, prohledáme pod ním oblast podobnou výše popsanému útvaru. Na každém následujícím řádku se musí v určité oblasti pod dříve nalezeným bodem nacházet alespoň jeden další bod. Podrobný popis je na

Obr. 3.3. Dokud tomu tak bude, budeme pokračovat dál. Až nenalezneme žádný další bod, prohlásíme dříve nalezené body za rovnou čáru a vhodně ji interpretujeme. Pokud možno tak, aby co nejlépe odpovídala své hranové předloze.

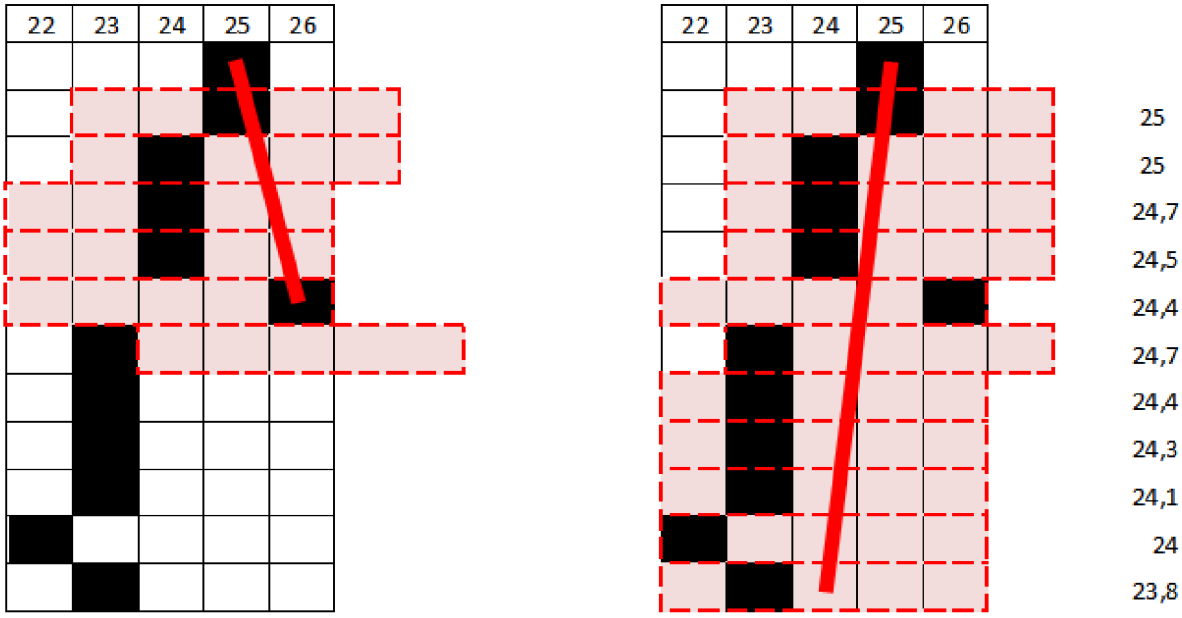


Obr. 3.3: Prohledávání oblasti pod již nalezeným bodem.

Problémem je nalezení toho správného výchozího bodu na dalším řádku. Pokud zůstaneme při posunu na další řádek v témže sloupci, ve kterém byl nalezen předchozí bod, bude hledání vystaveno velkým nepřesnostem z důvodu rozptylu jednotlivých bodů. V takovém případě bychom se nepohybovali ve výše zmíněném obdélníku, ale chaoticky bychom procházeli předem nedefinovanou oblast. To by bylo v rozporu s myšlenkou, že sledujeme předpokládaný směr hledané čáry.

Na Obr. 3.4 je taková situace znázorněna. V obou částech obrázku je naznačena stejná testovací čára. Tabulka představuje rastr obrázku, černá políčka reprezentují jednotlivé body. Každý sloupec je očíslován, v příkladu se tedy jedná o sloupce 22 až 26. Červeně podbarvená ohraničená oblast značí prostor, ve kterém předpokládáme, že se čára bude nacházet. První část obrázku ukazuje metodu, kdy je výchozí bod hledání na dalším řádku ve stejném sloupci, jako byl dříve nalezený bod. Je zřejmé, že se nepohybujeme nijak ustáleným směrem. Proto je čára detekována zcela chybně.

V druhé části obrázku je již prohledávaná oblast pravidelná a splňuje dané požadavky. Je toho docíleno výpočtem následujícího výchozího bodu z průměrné polohy dříve nalezených bodů. Čísla vpravo ukazují vypočítanou hodnotu výchozího bodu v každém řádku. Tím je dosaženo dodržení správného směru prohledávání. Odskoky jednotlivých bodů jsou díky tomu odfiltrovány a detekce je tak robustnější. V prvním případě způsobil jeden chybný bod předčasné ukončení čáry, protože na dalším řádku se v prohledávaném prostoru žádný jiný bod nenacházel. Metoda vycházející z průměrné polohy nalezených bodů tím byla ovlivněna pouze mírně, a proto nalezená čára odpovídá své předloze. Pokud je na následujícím řádku v prohledávané oblasti nalezeno více bodů, vybereme pouze ten, který je co nejblíže vypočtené výchozí hodnotě na daném řádku.



Obr. 3.4: Detekce svislé čáry dvěma různými způsoby.

Nevýhodou této metody je nemožnost detekce šikmých čar. Vzhledem k tomu, že do průměru jsou započítány všechny předchozí body, nemůže se čára příliš odklonit od svislé osy. U šikmých čar by algoritmus nedetekoval čáru v celé délce, ale jako několik krátkých čar vzájemně odsazených. Proto se při výpočtu nepracuje se všemi dříve nalezenými body, ale pouze s posledními několika. Díky tomu je možné detekovat i šikmé čáry. Metoda je tedy odolná vůči šumu, toleruje mírné zakřivení čáry a je relativně přesná. Při testování dosahuje mnohem přijatelnějších výsledků, než Houghova transformace. Chyby se projevují pouze v případě, že je fotografie rozmazaná nebo v případě, kdy dveře překrývá jiný předmět.

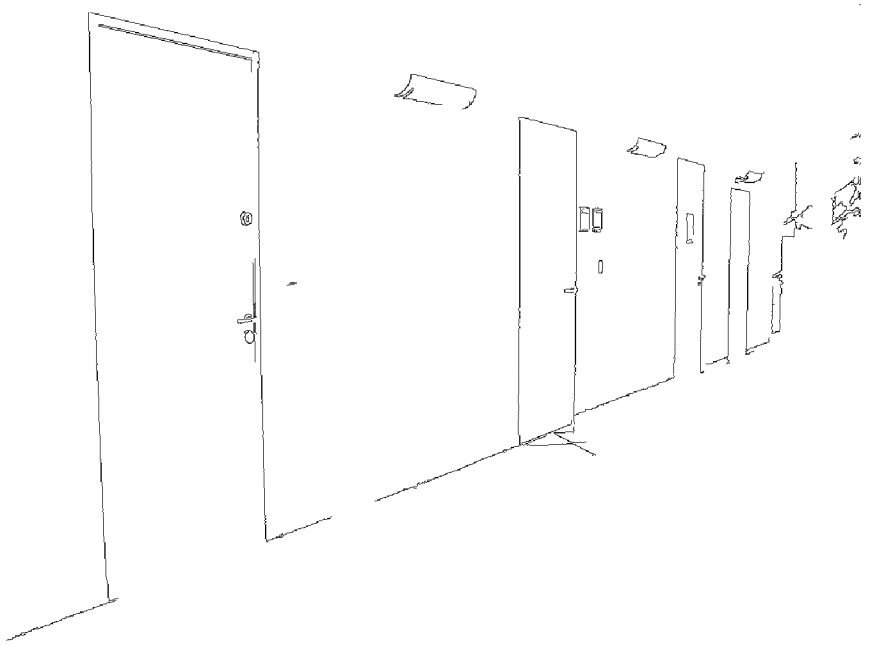
Detekce nesvislých čar

Při pokusech aplikovat stejný algoritmus na nesvislé čáry jsem neuspěl. V situaci, kdy se hledané čáry mohou ubírat všemi směry, je velmi obtížné zjišťovat, zda jsou čáry opravdu rovné. Je sice možné při nalezení hranového bodu prohledat jeho okolí a ukládat nalezené body, ale nezískáme tak podstatné informace o vzájemných vztazích mezi jednotlivými body. Jinak řečeno z množiny bodů těžko odhadneme, zda se jedná o rovnou čáru nebo jakýkoliv jiný útvar.

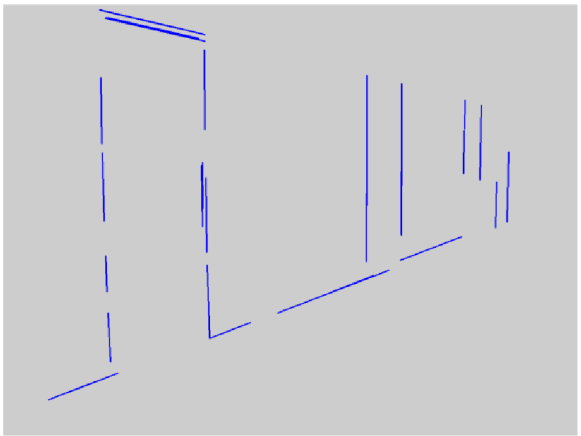
Při návrhu algoritmu pro detekci nesvislých čar jsem se rozhodl, že budu obrázek procházet vždy v jednom předem určeném směru, dokud budu nacházet nějaké body. Je zřejmé, že řadu nalezených bodů, které leží vedle sebe v určitém směru, lze prohlásit za rovnou čáru. Zbývá jen definovat ten směr. Houghova transformace tento problém řeší tak, že daným bodem postupně prokládá přímky. Každá další je oproti té předchozí natočena o předem daný úhel, třeba jeden stupeň. Ve svém algoritmu vypočítávám směr z existujících bodů, neprocházím zbytečně celý prostor. Přesněji v okolí daného bodu hledáme další body, které leží vedle sebe. Po určitém předem definovaném počtu nalezených bodů můžeme předpokládat, že máme potenciální počátek nové čáry. Předpokládejme třeba 8 bodů. Těmito body proložíme přímku. Pokud se opravdu jednalo o počátek čáry, bude ve směru přímky nebo v jejím blízkém okolí hledaná čára pokračovat a bude tedy možné nacházet v tomto směru další body.

Důležitým prvkem je neustálé zpřesňování vypočítaného směru hledání na základě polohy nově nalezených bodů. Za prvé je osm nalezených bodů pro přesné určení směru málo a za druhé nemáme jistotu, že všechny body byly opravdu součástí přímky. Výsledek také negativně ovlivňuje šum a jiné nepřesnosti v obraze. Proto při nalezení dalšího bodu znovu všemi body, i těmi dříve nalezenými, proložíme přímku a hledáme dál v jejím směru. Podrobněji bude algoritmus rozebrán v kapitole Implementace.

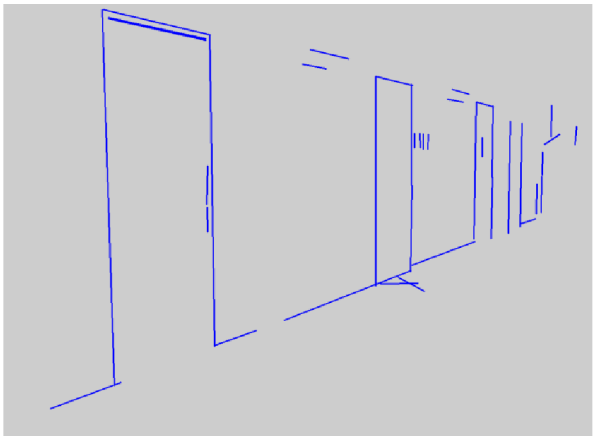
Na následujících obrázcích jsou rozdíly při detekci čar pomocí Houghovy transformace a výše uvedených algoritmů.



Obr. 3.5: Výstup Cannyho detektoru.



Obr. 3.6: Houghova transformace.



Obr. 3.7: Vlastní algoritmus detekce čar.

3.2 Vanishing Point

Pro detekci Vanishing pointu využijeme dříve nalezených nesvislých čar. Základním předpokladem je, že v největší míře budou v obrázku zastoupeny čáry, které směřují do Vanishing pointu. Čili většina předmětů na zdi umístěných vodorovně. Například obraz, ozdobné obložení, nástěnka, nábytek stojící u zdi a hlavně hrany dveří. Všechny hrany, kterými jsou tyto předměty určeny, je sbíhají v jednom pomyslném bodě, kterému říkáme Vanishing point. Dobrý způsob, jak tyto významné čáry oddělit od těch ostatních, je algoritmus RANSAC [1].

Hlavním cílem algoritmu je oddělit významná data od těch nevýznamných, na základě informací o tom, jak má výsledek vypadat a jaké vztahy by měly být mezi jednotlivými prvky, které výsledek určují. V případě Vanishing pointu se snažíme najít pokud možno co největší počet čar, které směřují do jednoho bodu. K tomuto účelu jsem vytvořil dva algoritmy založené na RANSACu. Oba pracují následujícím způsobem:

1. Výběr dvojice čar z množiny nesvislých čar (náhodně nebo systematicky).
2. Výpočet průsečíku těchto čar – Vanishing point.
3. Zjištění ohodnocení Vanishing pointu podle polohy ostatních čar z množiny.
4. Výsledkem je Vanishing point s nejvyšším hodnocením.

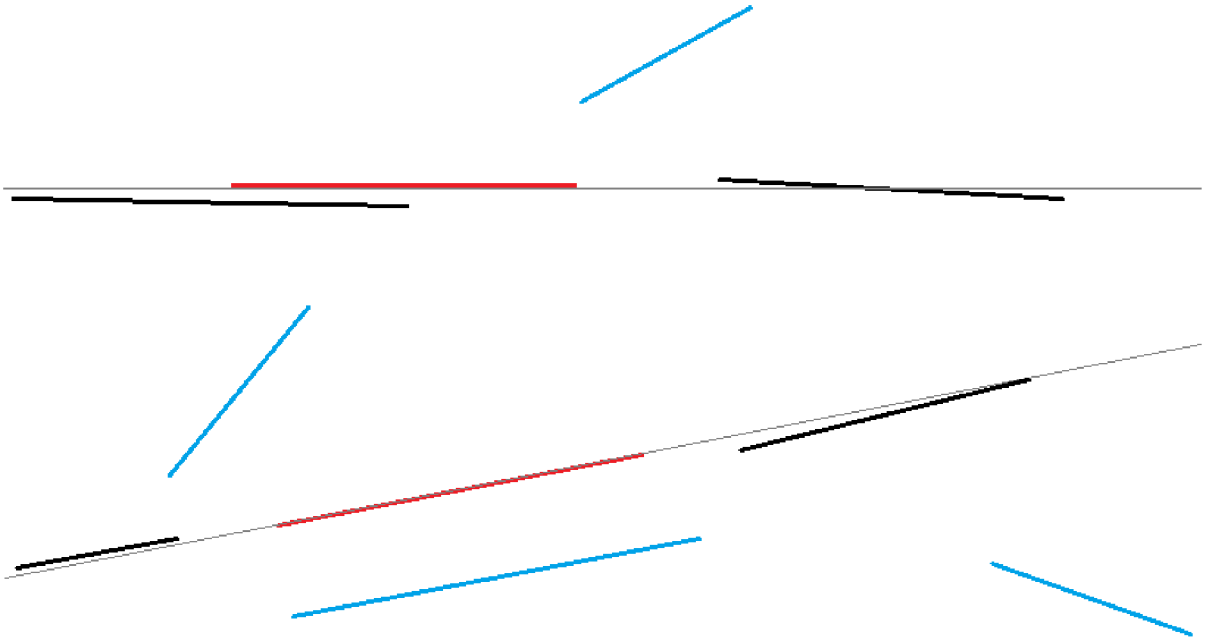
Algoritmy se vzájemně liší v bodě 1 a 3.

Algoritmy pro výběr Vanishing pointu

První metoda začíná náhodným výběrem dvojice čar. Průsečík těchto čar nechť určuje Vanishing point. Pokud bychom opravdu vybrali čáry, o kterých toto tvrzení platí, bylo by v obrázku mnoho jiných čar, které budou ležet v jejich blízkém okolí. Přesněji byly by rovnoběžné s vybranými čarami a ležely by v jejich blízkosti.

Nad množinou nesvislých čar je výběr náhodné dvojice čar proveden vícekrát. Dále v textu budu tyto čáry označovat jako „vybrané“. Nejdřív ověříme, zda vybraná dvojice čar svírá úhel, který by mohl odpovídat čarám určujícím Vanishing point. Předpokládejme třeba 10 a 35 stupňů. Dále zjistíme vzdálenost všech ostatních čar od každé z vybraných čar a úhel, který spolu svírají. Pokud budou tyto hodnoty v patřičných mezích, budeme čáry považovat za blízké. Ta dvojice, která bude mít nejvíce blízkých čar, bude prohlášena za vítěze a průsečík těchto čar bude Vanishing point. Při výpočtu vzdálenosti jsou obě vybrané čáry považovány za přímky. Všechny ostatní čáry, které s nimi porovnáváme, jsou zjednodušeny na jediný bod, kterým je jejich střed. Počítáme tedy vzdálenost bodu od přímky. Takto získaná hodnota je relevantní, protože vzdálenost je počítána pouze u těch čar, které s jednou z vybraných čar svírají dostatečně malý úhel na to, aby tyto dvě čáry byly považovány za rovnoběžné.

Na Obr. 3.8 je příklad dvou vybraných čar. Jsou červené a pro názornost jimi prochází přímky. Černé čáry byly vyhodnoceny jako blízké, modré ne.



Obr. 3.8: Příklad blízkých čar podle prvního algoritmu.

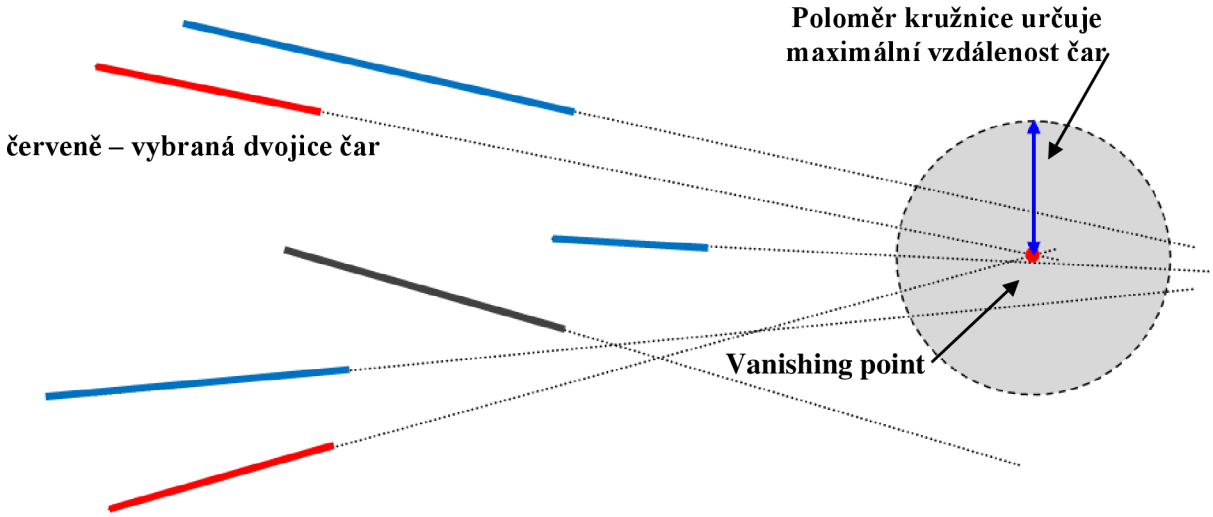
Z obrázku je zřejmé, že se může objevit mnoho čar, které budou do Vanishing pointu směřovat, ale nebudou započítány. Což může negativně ovlivnit výsledek. Příkladem je modrá čára vlevo dole, která je od červené čáry dál, než dovoluje daný parametr. Přesto se tato čára na definici Vanishing pointu podílí. Druhým úskalím algoritmu je náhodný výběr čar. Správný výsledek je závislý na tom, zda bude vybrána ta pravá dvojice čar, která Vanishing point nejlépe definuje. Navíc je výsledek vlivem náhodného výběru pokaždé jiný. Vzhledem k těmto vlastnostem není algoritmus příliš spolehlivý, proto jsem vytvořil druhý, jehož výsledky jsou mnohem lepší.

Druhý algoritmus neprochází množinu nesvislých čar náhodně, ale systematicky vybírá všechny možné dvojice, nad nimiž provádí výpočty. Každá testovaná dvojice čar obdrží bodové hodnocení v závislosti na tom, v jakém vztahu je s ostatními čárami. Výsledkem je ta dvojice, jejíž hodnocení je nejvyšší.

V první fázi je spočítán průsečík vybrané dvojice čar. Ve druhé fázi je spočítáno hodnocení pro tento průsečík a dvojici čar. Hodnocení je tím vyšší, čím více čar směřuje do nalezeného bodu. Na rozdíl od předchozí metody není brána v úvahu vlastní poloha vybrané dvojice čar, ale pouze poloha jejich průsečíku. Díky tomu se do ohodnocení mohou započítat všechny čáry, které do nalezeného bodu směřují a ne pouze ty, které jsou poblíž vybraných čar.

Navíc je zohledněna i délka každé čáry. Čím delší čára, tím vyšší ohodnocení. Tím je zajištěno, že dlouhé čáry, které se objevují třeba jako hrany mezi podlahou a stěnou, budou mít na výsledek podstatně větší vliv, než krátké hrany různých předmětů. Hodnocení vybraných dvou čar je tedy spočítáno jako součet dílčích ohodnocení, které získaly ostatní čáry směřující do jejich průsečíku. V úvahu bereme pouze ty čáry, které jsou k průsečíku dostatečně blízko. Na příklad vezměme hodnotu 40 bodů. Na první pohled se může tato vzdálenost jevit jako relativně velká. Z testování však vyplývá, že vlivem perspektivy, špatného natočení fotoaparátu nebo nepřesností objektivu lze pozorovat mnoho čar, které do Vanishing pointu reálně směřují, ale přesný výpočet odhaluje velkou odchylku. V hodnocení se větší vzdálenost promítne menší výslednou hodnotou, což sníží zkreslení případnými rušivými čarami, které do Vanishing pointu přímo nesměřují. Schematický nákres je na Obr. 3.9. Červený bod vpravo je Vanishing point, který byl vypočten jako průsečík

červených čar. Modré čáry směřují do pomyslné kružnice, která znázorňuje maximální vzdálenost od Vanishing pointu. U těchto čar bude počítáno hodnocení. Černá čára směřuje jinam, proto započítána nebude.



Obr. 3.9: Schematické znázornění způsobu hodnocení čar ve druhém algoritmu.

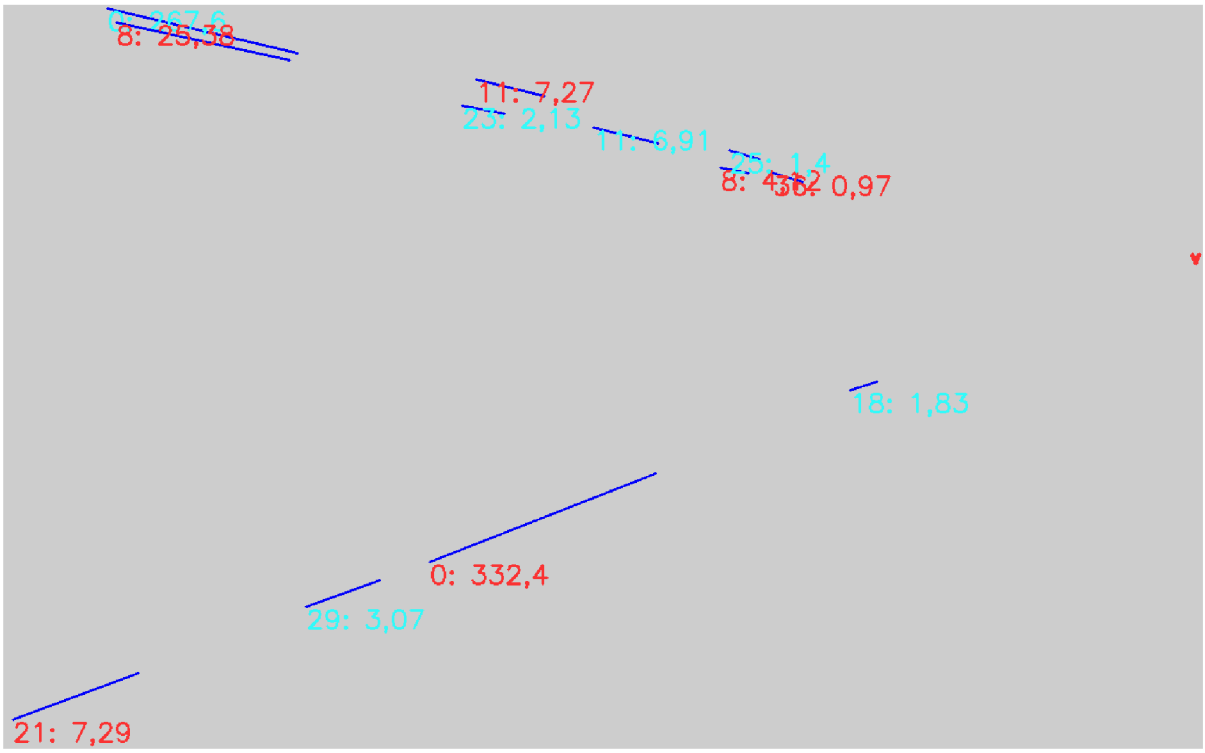
Celkové hodnocení dvojice čar a jejich průsečíku, je dáno součtem všech dílčích hodnocení. Za Vanishing point je prohlášen ten průsečík, který má nejvyšší hodnocení. Přesný vztah pro výpočet dílčího hodnocení každé čáry, která směřuje do blízkosti Vanishing pointu udává rovnice číslo 1, kde *D* je délka čáry, *Kd* je koeficient délky, *V* je vzdálenost čáry od Vanishing pointu a *Kv* je koeficient vzdálenosti. Hodnoty obou koeficientů umožňují ve výpočtu libovolně zvýhodnit či znevýhodnit délku čáry i vzdálenost od Vanishing pointu. Můžeme tak ovlivnit, jestli pro nás budou důležitější delší čáry nebo bližší čáry. Podrobně budou oba parametry vysvětleny v kapitole Implementace. Vzdálenost čáry je počítána jako vzdálenost Vanishing pointu od přímky, která prochází danou čarou. Převrácená hodnota vzdálenosti je ve vzorci proto, aby bylo hodnocení tím vyšší, čím je čára blíže. Pokud je vzdálenost rovna nule, čára tedy směřuje přímo do Vanishing pointu, použije se rovnice číslo 2. Místo převrácené hodnoty vzdálenosti je brána hodnota 1,2. Tím je zajištěno, že tato čára bude mít větší hodnocení, než čára vzdálená jeden bod od Vanishing pointu.

$$Hodnocení = D^{Kd} * V^{-Kv} \quad (1)$$

$$Hodnocení = D^{Kd} * 1{,}2 \quad (2)$$

Příklad výstupu této metody je na Obr. 3.10. Vanishing point označuje písmeno V vpravo. V popisku každé čáry je nejdříve vzdálenost od Vanishing pointu a za dvojtečkou hodnocení. Oba koeficienty jsou v příkladu nastaveny na hodnotu 1. Znamená to stejnou váhu dílky i vzdálenosti. To však nemusí být vždy ideální řešení. Je zřejmé, že hodnocení je v příkladu nejvíce závislé na vzdálenosti čar od Vanishing pointu. Například dvě čáry vlevo nahoře. Jedna z nich určuje zárubně dveří, druhá horná hranu dveří. Obě čáry by tedy měly směřovat vlivem perspektivy do stejného bodu, protože jsou rovnoběžné. Mírný sklon způsobený výše popsanými vlastnostmi každého snímku zapříčinil, že druhá čára má desetkrát menší hodnocení, než ta první. Proto bývá v praxi dobré

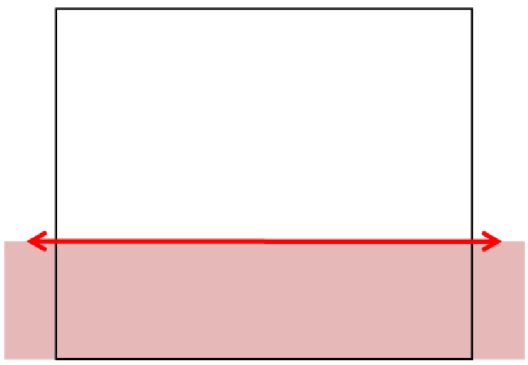
nastavit koeficient délky na vyšší hodnotu, třeba 2 a koeficient vzdálenosti na nižší hodnotu, třeba 0,5.



Obr. 3.10: Hodnocení čar.

Velkým problémem při hledání Vanishing pointu je dlažba. Velké dlaždice vytváří snadno detekovatelnou síť hran, které směřují do jednoho bodu, většinou uprostřed obrazu. Správná poloha Vanishing pointu je ale vždy mimo obraz v horní polovině. Jedinou výjimkou je případ, kdy jsou na snímku obě stěny chodby. Potom může být Vanishing point v obraze. Avšak i v tomto případě je vždy v horní části obrázku, protože ve spodní části je podlaha.

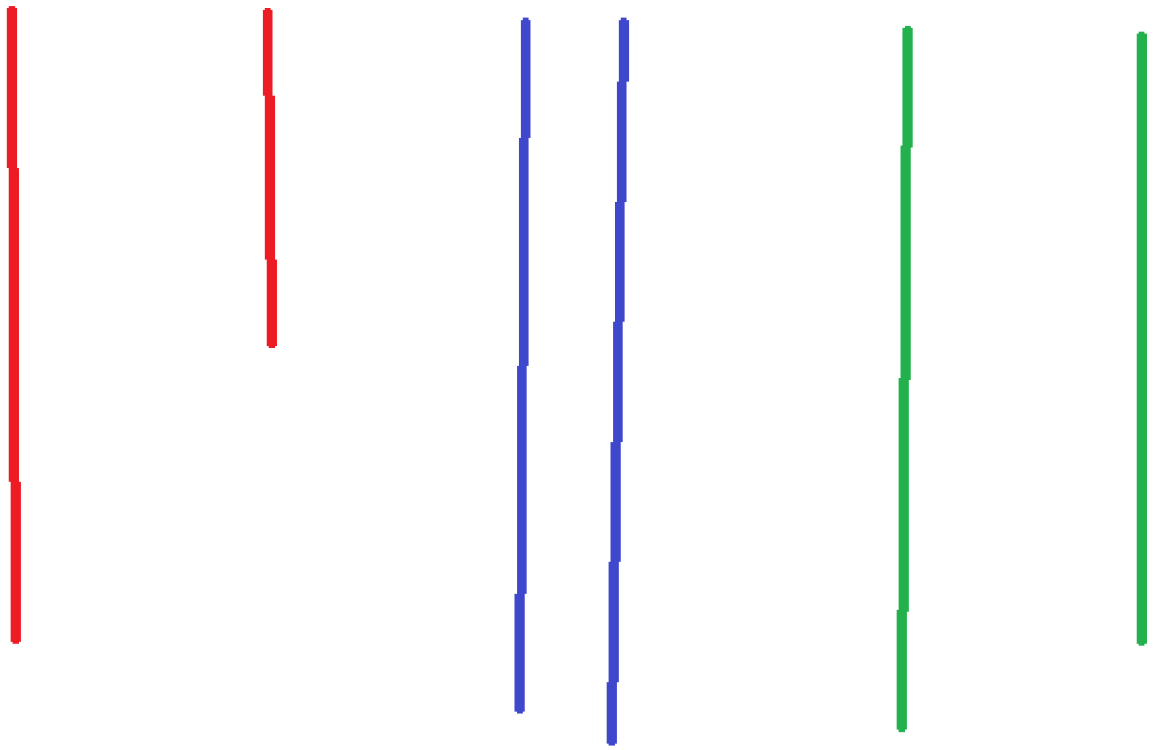
Je tedy zřejmé, že pokud najdeme Vanishing point ve spodní části obrazu, jedná se o chybnou detekci. Takový výsledek je zahozen. Za spodní hranici považuji poslední třetinu obrázku. Toto pravidlo platí i mimo obraz. Na Obr. 3.11 je tato hranice znázorněna. Červené podbarvení značí zakázaný prostor. Šipky znázorňují, že se jedná i o oblast mimo obrázek. Díky tomuto opatření je algoritmus mnohem úspěšnější.



Obr. 3.11: Místo, kde se Vanishing point nemůže nacházet.

3.3 Označení dveří

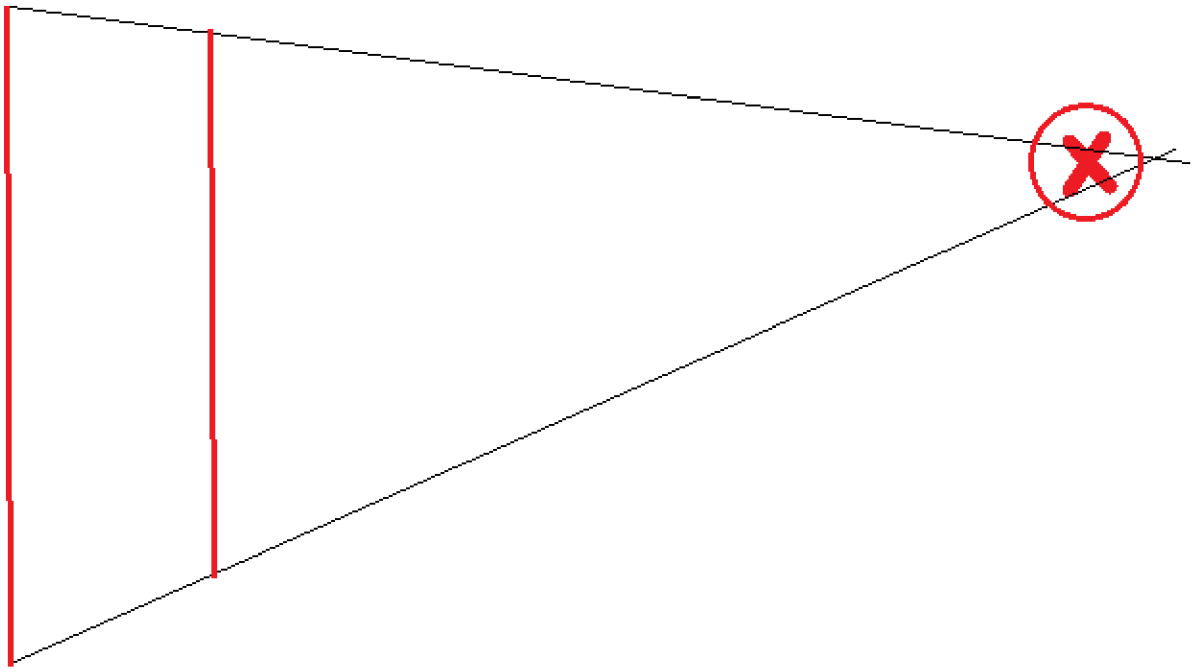
S využitím dříve získaných informací bude provedena detekce dveří. K dispozici máme množinu svislých čar, množinu nesvislých čar, Vanishing point a samozřejmě vstupní obraz. K základnímu vymezení kandidátů použijeme svislé čáry. Dveře mohou definovat pouze ty čáry, které si odpovídají svoji výškou, a jejich vzájemná vzdálenost odpovídá určité hodnotě. Ze všech svislých čar tedy vybereme pouze ty dvojice, které splní dané parametry. Na Obr. 3.12 Jsou znázorněny tři dvojice čar. V červené dvojici si čáry neodpovídají délkou, modré čáry jsou příliš blízko. Zelená dvojice vyhovuje z pohledu vzájemné vzdálenosti i délky čar. Proto je tato dvojice označena jako kandidát na dveře. Rovnoběžnost čar není třeba řešit, protože procházíme pouze množinu svislých čar.



Obr. 3.12: Dvojice čar, které mohou označovat dveře. Pouze zelená dvojice vyhovuje.

Na vybrané kandidáty budou aplikovány další metody, které potvrdí, zda se opravdu jedná o dveře či nikoliv. Výsledný verdikt na základě výsledků těchto metod je ponechán na uživateli. První metoda zjišťuje, zda se v blízkosti hraničních bodů vybraných svislých čar nachází i čáry nesvislé. Hledá spojnice mezi horními nebo spodními body svislých čar. Pokud existují obě spojnice, je velmi pravděpodobné, že se jedná o dveře.

Dalším příznakem je zjištění, zda se dveře nachází na stěně. Pokud horní i spodní hrana dveří bude směřovat do Vanishing pointu, můžeme předpokládat, že dveře leží na stěně. V tomto případě nebudeme brát v potaz, zda byly tyto hrany v předchozím kroku nalezeny či nikoliv. Za horní a spodní hranu dveří budeme považovat ideální spojnice svislých čar v jejich koncových bodech. Obě metody tak mohou být provedeny nezávisle na sobě. Na Obr. 3.13 je znázorněno protažení těchto hran směrem k Vanishing pointu.

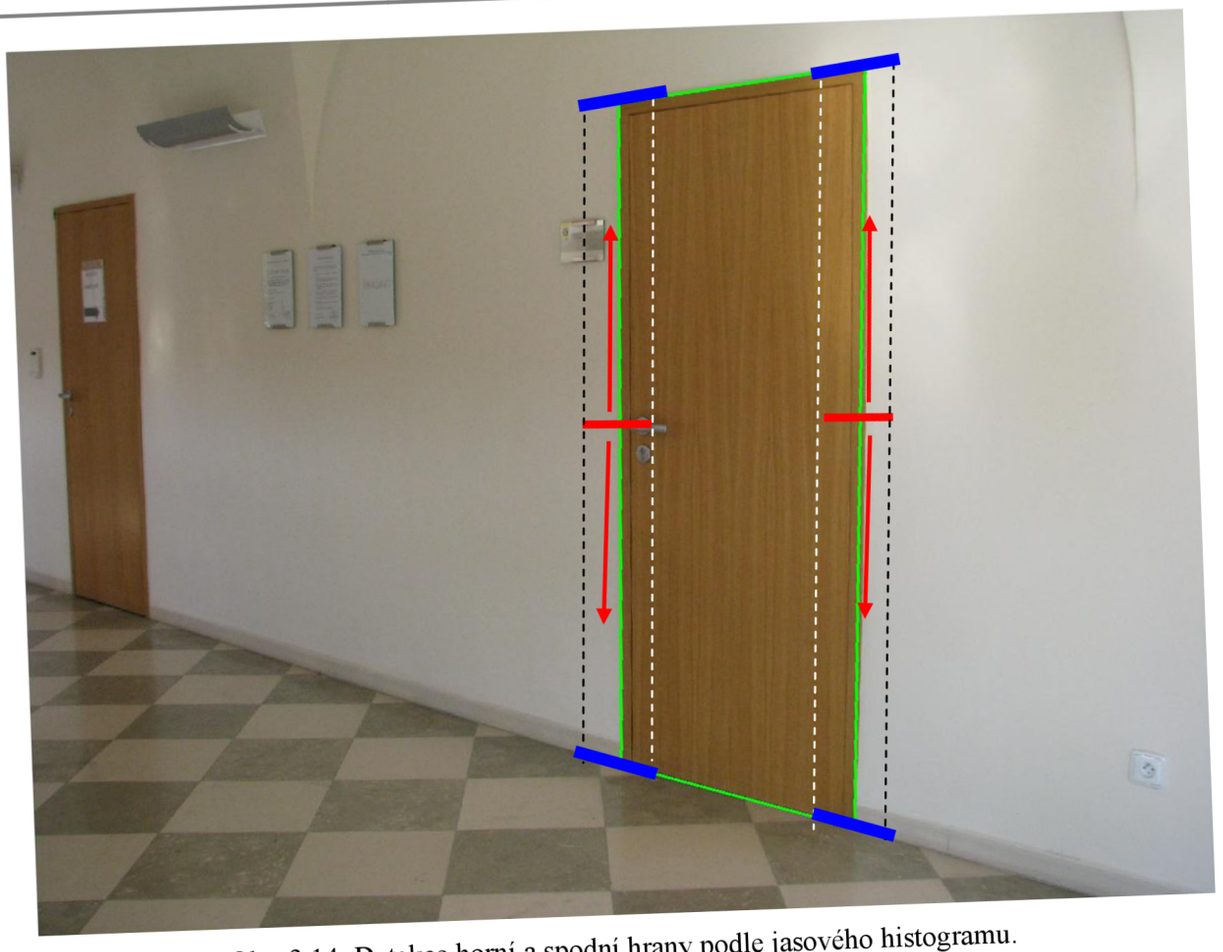


Obr. 3.13: Zjištění, zda se dveře nachází na stěně.

Červený kříž značí Vanishing point, kružnice kolem něj určuje maximální vzdálenost horní a spodní hrany dveří od Vanishing pointu. Pokud obě hrany leží v kružnici, předpokládáme, že dveře jsou na stěně.

V posledním testu vycházíme z toho, že dveře mají ve většině případů jinou barvu, než jejich okolí. Pokud budeme sledovat jasový histogram podél bočních hran dveří, mělo být možné určit na základě podobnosti histogramů horní a spodní hranici dveří. Jako počátek vezměme výškový střed dveří, od kterého se budeme pohybovat nahoru a dolů podél obou svislých hran. Postupně budeme počítat histogramy. Pokud se pohybujeme v rámci dveří, neměly by se sousední histogramy vzájemně lišit. V případě, že překročíme hranici dveří, rozdíl mezi histogramy bude náhle velmi významný. Podle toho lze usoudit, kde dveře začínají a končí. Na Obr. 3.14 je konkrétní příklad se znázorněním postupu. Červené šipky znázorňují směr postupu. Začínáme uprostřed. Zvolíme útvar, jehož šířka bude odpovídat vzdálenosti černé čáry od bílé a výška bude několik málo bodů. Z této části obrázku spočítáme jasový histogram. Posuneme se v naznačeném směru o několik bodů a stejným způsobem výpočet opakujeme. Korelací dvou sousedních histogramů zjistíme, jak moc se od sebe liší. Pokud se jedná o histogramy, které byly spočítány v místě, dveří, neměly by se od sebe příliš lišit. Na hranicích dveří znázorněných modře bude naopak mezi histogramy velký rozdíl. Této vlastnosti využijeme při označení horní a spodní hrany dveří. Detekce hran na základě rozdílů mezi jasovými histogramy poskytuje v kombinaci s metodou hledání příslušných nesvislých čar dobrý důkaz, že se skutečně jedná o dveře.

Pro výsledné označení dveří můžeme využít všech tří dříve zmíněných příznaků. Na uživateli zbývá rozhodnout, zda požaduje přísnou detekci, tedy za dveře označí pouze ty kandidáty, kteří úspěšně vyhoví všem kontrolním metodám, nebo zda na základě vhodné kombinace těchto metod vybere i kandidáty, kteří uspěli pouze částečně.



Obr. 3.14: Detekce horní a spodní hrany podle jasového histogramu.

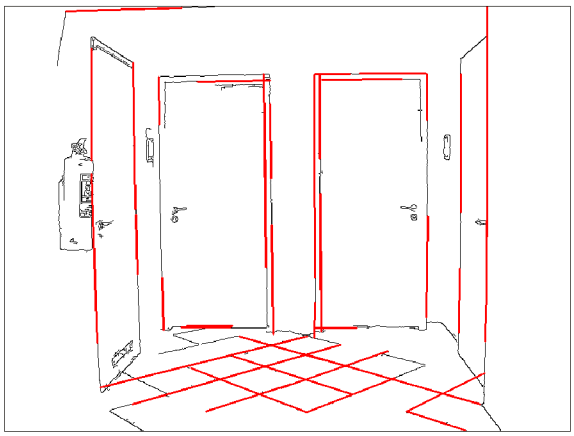
4 IMPLEMENTACE

Aplikace je napsána v jazyce C++ s použitím knihovny OpenCV. CV znamená „Computer vision“, jedná se o knihovnu, která slouží pro práci s grafikou. Umožňuje různé manipulace s obrázky, poskytuje datové typy pro snadnou práci s body a čarami a mnoho dalšího. Dále v textu budou popsány ty části knihovny, se kterými jsem pracoval.

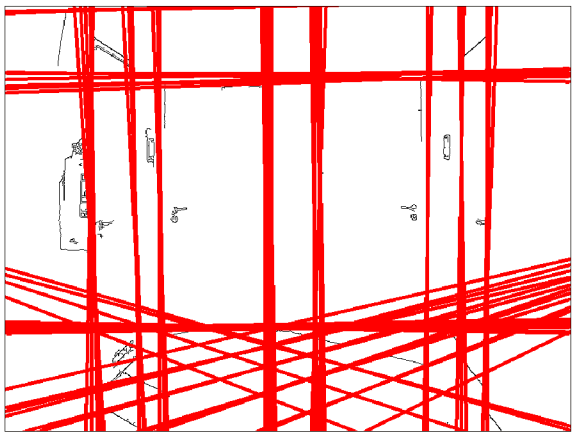
Program pro detekci dveří tvoří jedna třída, která zapouzdřuje veškerou funkčnost a umožňuje tak snadné použití v jiném programu nebo v robotovi. Uživatelské rozhraní je cíleno na operační systém Microsoft Windows. Jedná se o jednoduchý formulář, který umožňuje nastavení parametrů detekce a její následné spuštění na vstupním obrázku.

4.1 Problémy s rovnými čarami

V návrhu byla zmíněna metoda pro detekci rovných čar – Houghova transformace. V knihovně OpenCV jsou k dispozici dvě funkce, které ji implementují. *HoughLines*, která v obraze vyhledá úsečky, a *HoughLinesP*, která umí detekovat přímky. Na Obr. 4.1 a Obr. 4.2 jsou ukázány výstupy obou funkcí vedle sebe. Černé čáry jsou výstupem Cannyho detektoru, červené Houghovy transformace.

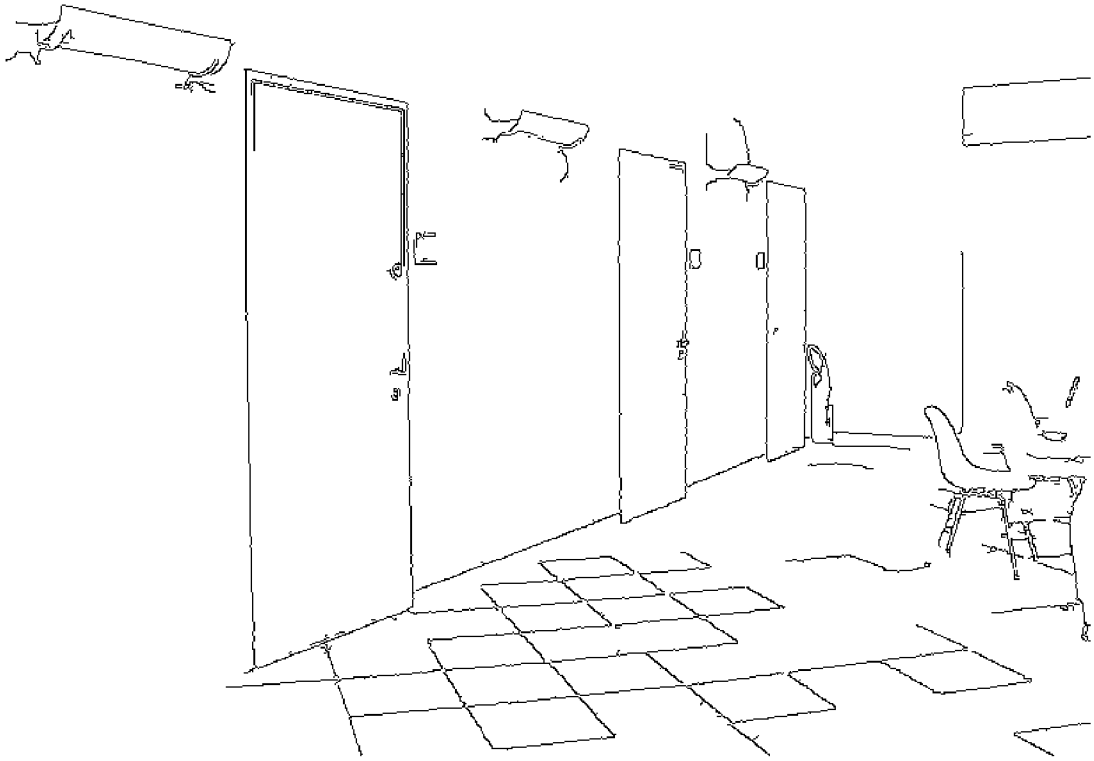


Obr. 4.1: Výstup funkce Probabilistic Hough transform – úsečky.

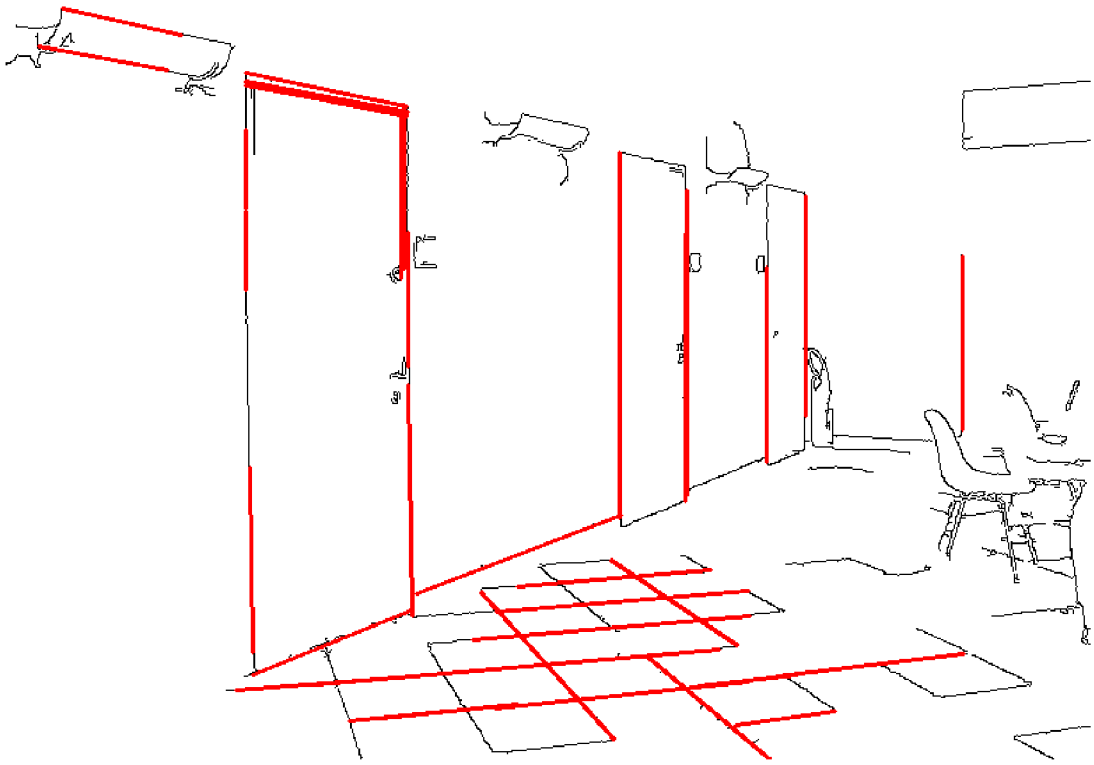


Obr. 4.2: Výstup funkce Hough Transform – přímky.

V programu jsem plánoval použít funkci *HoughLinesP*. Přesto, že je možné funkci nastavit pomocí několika parametrů, nebylo možné s její pomocí získat souvislé přímky. Velmi zřídka byla vstupní čára, byť relativně rovná, detekována celá. Na následujících obrázcích je nejdříve hranová reprezentace vstupního obrazu vytvořená Cannyho detektorem. Další obrázek obsahuje to stejné a je doplněný o čáry, které byly nalezeny funkcí *HoughLinesP*. Z pozorování tohoto a mnohých dalších obrázků usuzuji, že funkci *HoughLinesP* nejvíce vadí zahnuté hrany. Což je vlastnost, kterou vykazují všechny fotografie. Dobře patrné je to například na prvních dveřích, kde je levá hrana v dolní části zahnutá a proto detekce dopadla špatně.

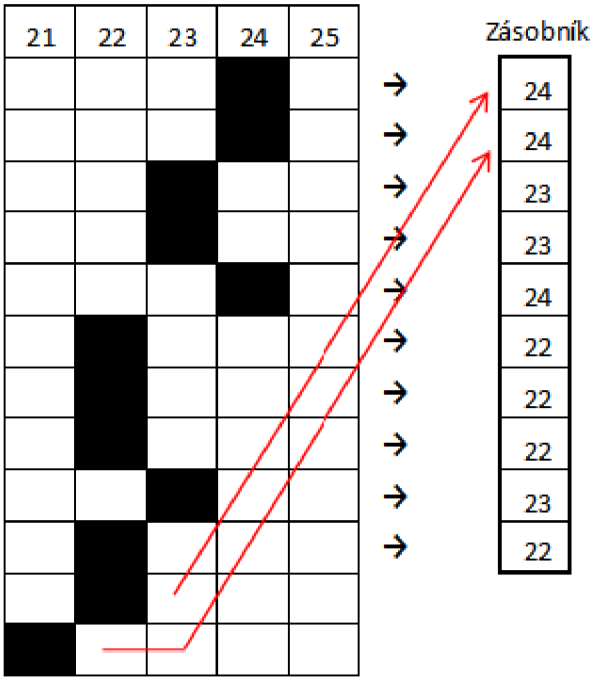


Obr. 4.3: Hranová reprezentace obrázku vytvořená Cannyho detektorem.



Obr. 4.4: Čáry nalezené funkcí HoughLinesP.

Především tento nedostatek jsem se snažil vyřešit vlastním algoritmem pro detekci svislých čar. Při hledání procházíme obrázek bod po bodu a hledáme hranu. Obrázkem myslím jeho hranovou reprezentaci, která byla vytvořena Cannyho detektorem. Při nalezení hranového bodu voláme metodu *HledejDalsiDole()*, která najde případnou čáru vycházející z tohoto bodu. Metoda se postupně po řádcích posouvá dolů a prohledává danou oblast pod dříve nalezeným bodem. Tuto oblast je možné ovlivnit následujícími parametry: *caraMaxPosun*, který nastavuje velikost prohledávaného prostoru na dalším řádku (viz Obr. 3.3: Prohledávání oblasti pod již nalezeným bodem.), *caraPocetZohlednovanychPixelu*, který nastavuje počet bodů, z nichž se vypočítává výchozí poloha pro hledání na dalším řádku a *caraMinDelka*, který nastavuje minimální délku nalezené čáry. Parametr *caraPocetZohlednovanychPixelu* definuje velikost zásobníku, do kterého jsou ukládány polohy nalezených bodů. Přesněji čísla sloupců, kde byly body nalezeny. Z těchto hodnot vypočítáme průměr a dostaneme číslo sloupce, který považujeme za výchozí pro hledání na následujícím řádku. Na Obr. 4.5 je znázorněno postupné ukládání hodnot do zásobníku. Průměr z hodnot prvních deseti bodů je 22,9, čili sloupec 23. Je však zřejmé, že čára bude pokračovat vlevo a proto se tato hodnota stává neaktuální. Při nahrazení prvních dvou záznamů polohami nových bodů je průměr 22,4, díky čemuž se stane výchozím sloupec 22.



Obr. 4.5: Nahrazování starších hodnot v zásobníku novými.

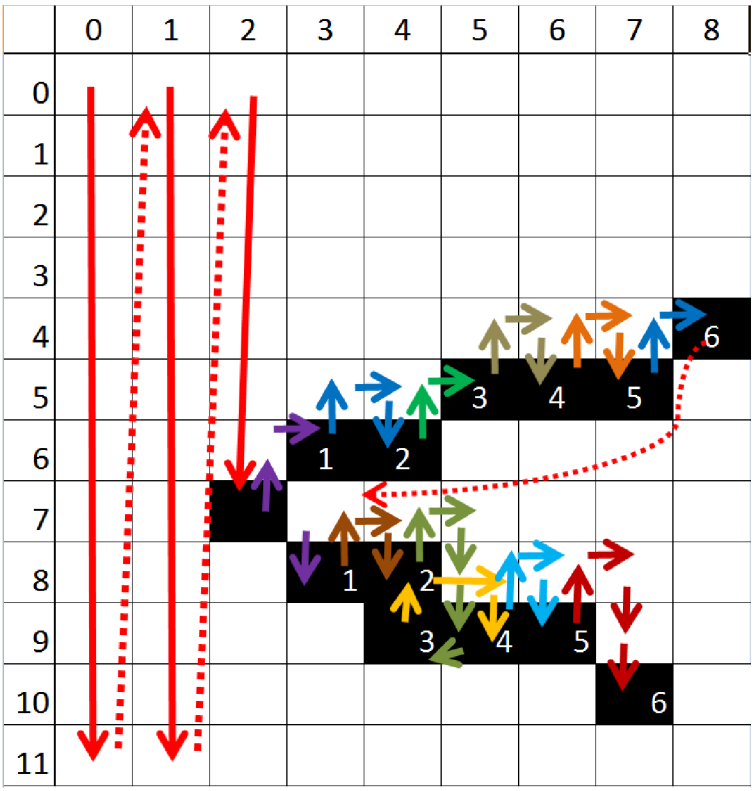
Při testování se jako optimální ukázalo nastavení parametru *caraPocetZohlednovanychPixelu* na hodnotu 10 a *caraMaxPosun* na hodnotu 2. Nastavení minimální délky čáry závisí na rozlišení vstupního obrazu. Při rozlišení 800x600 obrazových bodů jsem používal hodnotu 40.

Každý nalezený bod je označen, aby nebyl znovu započítán. Pokud v prohledávané oblasti nic dalšího nenajdeme a detekovaná čára má potřebnou délku, ukončíme hledání a vracíme výsledek. Za počátek čáry je považována průměrná hodnota z prvního naplněného zásobníku, tedy z prvních deseti bodů. Poslední hodnotou je průměr posledních deseti bodů. Jedná se o čísla sloupců, pro čísla řádků se přirozeně použije první a poslední nalezený bod. V praxi se metoda osvědčila dobře, výsledek kopíruje svoji předlohu.

U detekce nesvislých čar je největším problémem to, že neznáme směr čáry. Svislé čáry směřovaly kolmo dolů, kdežto ty ostatní mohou mířit kterýmkoliv směrem. V první řadě je nutné procházet obrázek po sloupcích, aby hledané čáry ležely napříč. Dále je nutné najít několik prvních bodů čáry a z nich určit směr, ve kterém budeme dále hledat.

Pro nalezení prvních několika bodů čáry slouží funkce *RekurzivniHledani()*. Prohledá pravé okolí prvního bodu – viz Obr. 4.6. U prvního nalezeného bodu prohledá opět jeho pravé okolí a tak rekurzivně postupuje, dokud nenajde zadaný počet bodu, pro příklad šest. Pokud není v daném směru nalezeno alespoň šest bodů, vrací se funkce o krok zpět a hledá dále. Počet bodů je možné nastavit parametrem *caraMaxPocetNalezenych*.

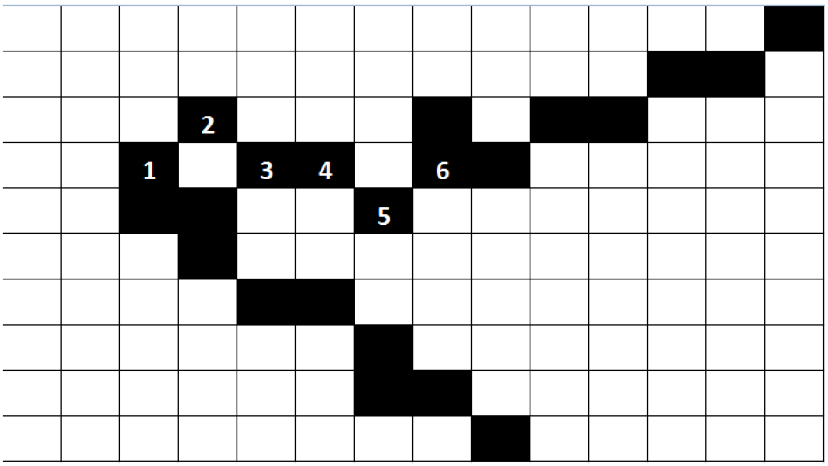
Červené čáry značí průchod maticí. První bod je nalezen na souřadnici 7:2. Rekurzivní hledání postupuje vždy nahoru, doprava, dolů, dolů, a doleva. Na souřadnici 6:3 je nalezen další bod, prohledává se tedy jeho okolí. Opět nahoru, doprava, dolů – zde nacházíme další bod. Takto postupujeme, dokud nacházíme další body. Až nalezneme šest bodů, zavoláme funkci *PocitejSmer()*, která bude dále pokračovat v detekci směrem, který vypočítá z nalezených šesti bodů. Funkce *RekurzivniHledani* každý nalezený bod označí, aby ho nezapočítala znovu, a dokončí prohledání okolí každého z těchto bodů. Tímto se dostaneme až k souřadnici 8:3, kde nacházíme další bod. Opět hledáme, dokud nemáme alespoň šest bodů. Pokud najdeme bodů méně a další postup již není možný, zahodíme poslední „slepý“ bod a vracíme se o úroveň zpět.



Obr. 4.6: Rekurzivní prohledávání pravého okolí bodů.

Pokud nalezneme šest po sobě jdoucích bodů, předáme jejich souřadnice metodě *PocitejSmer*. Jejím úkolem je pomocí metody nejmenších čtverců proložit těmito body přímku a podél ní hledat další body. V obrázku se posune vpravo do sloupce, který následuje za posledním nalezeným bodem. Pomocí získané rovnice přímky spočítá, kde přesně protíná přímka tento sloupec. Tím získáme přesné souřadnice dalšího bodu. Toto místo je prohledáno v okruhu dvou bodů ve všech směrech. Pokud je

v tomto prostoru nalezen nějaký bod, je považován za součást hledané čáry. Takto postupujeme dál, dokud nacházíme nějaké body. Je ale nutné stále zpřesňovat směr, ve kterém hledáme, protože není jisté, zda všechny body, obzvláště ty první, opravdu pochází z hledané hrany. Mohlo se také jednat pouze o náhodný shluk bodů způsobený šumem. Případně jde o místo, ze kterého vychází více čar. Příklad takové situace je na Obr. 4.7.



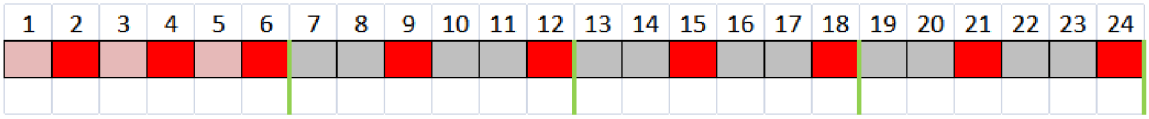
Obr. 4.7: Shluk bodů se dvěma vycházejícími čarami.

Prvním určením směru jsme získali pouze výchozí pozici, abychom nehledali úplně naslepo. Je však nutné směr prohledávání neustále přizpůsobovat nově nalezeným bodům. Pokud chceme do množiny dříve nalezených bodů přidat nové, musíme vše přepočítat se všemi body dohromady. To je dáno podstatou metody nejmenších čtverců. Takový výpočet ale není úplně triviální a jeho provádění při každém nově nalezeném bodu by mohl negativně ovlivnit rychlost celé detekce. Proto pokud je algoritmus provozován na pomalejším zařízení, je možné jej zjednodušit pomocí tří parametrů. První se jmenuje *smerPomerVyznamnychBodu* a určuje, které body budou do přepočtu zahrnuty. Pokud nastavíme hodnotu na 1, budou započteny všechny nalezené body. Při nastavení čísla 2 nebo 3 bude započten pouze každý druhý, respektive třetí nalezený bod. Pokud například nalezneme celkem 25 bodů a parametr nastavíme na hodnotu 3, započte se pouze 8 bodů. Tím se výrazně sníží náročnost, ale také přesnost výpočtu. Druhý parametr s názvem *smerPocetBoduProPrepocet* udává, kolik významných bodů je třeba nalézt, aby se provedl znovu výpočet metodou nejmenších čtverců. Pokud budeme brát v úvahu dříve zmíněný příklad doplněný o tento parametr s hodnotou 3, znamená to, že se bude přepočítávat pouze dvakrát. Jinými slovy parametr *smerPomerVyznamnychBodu* definuje filtr, který propouští pouze každý n-tý bod. Parametr *smerPocetBoduProPrepocet* určuje velikost zásobníku, do kterého se propuštěné, tedy významné body uloží. Až je zásobník plný, převedou se hodnoty do jiného zásobníků, který obsahuje dříve nalezené body. Přepočet se provede s použitím všech hodnot v zásobníku. Je tedy zřejmé, že každé další počítání se složitější, protože pracuje s více hodnotami.

Posledním parametrem je *smerPrvniPrepocet*, který má stejnou funkci jako parametr *smerPocetBoduProPrepocet*. Rozdíl je v tom, že *smerPrvniPrepocet* se použije pouze poprvé. To umožňuje určit počátek čáry poměrně přesně, i v případě, že potřebujeme algoritmus co nejvíce zjednodušit prvními dvěma parametry. Navíc je tím zpřesněna detekce krátkých čar.

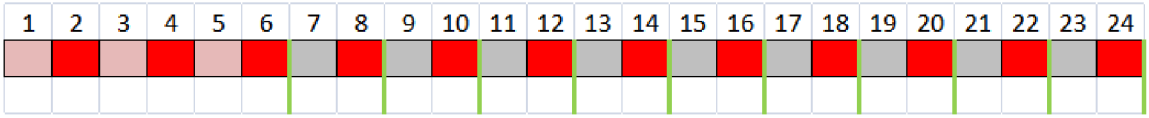
Body, které byly nalezeny funkcí *RekurzivniHledani*, jsou do zásobníku vloženy také. Nejsou ale použity všechny, protože by mohly negativně ovlivňovat výpočet. Počátek čáry bývá nepřesný působením šumu či zakřivení obrazu. V úvahu je tedy brán pouze každý druhý bod.

Smyslem těchto parametrů je výpočet co nejvíce zjednodušit, ale zachovat co největší přesnost. Význam parametrů bude lépe pochopitelný z Obr. 4.8. Prvních šest bodů bylo nalezeno funkcí *RekurzivniHledani*. Ostatní našla funkce *PocitejSmer*. Významné body jsou označeny červeně. Jsou to ty, které jsou uloženy do zásobníku. Svislé zelené čáry znamenají nový výpočet metodou nejmenších čtverců s použitím všech dat, které jsou zrovna v zásobníku. Šedé (růžové v případě prvních šesti) body nejsou zaznamenávány.



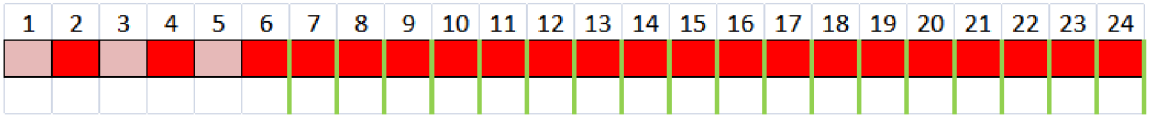
caraMaxPocetNalezenych: 6
smerPomerVyznamnychBodu: 3
smerPocetBoduProPrepocet: 2
smerPrvniPrepocet: 2

Ze 24 bodů je započítáno 9, výpočet je proveden 4-krát, vždy po šesti nalezených bodech. Toto nastavení považuji za optimální. Ušetří mnoho počítání a výsledek zůstává relativně přesný.



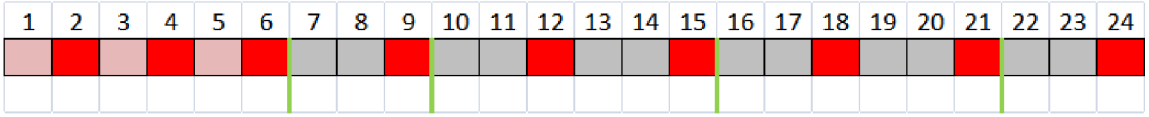
caraMaxPocetNalezenych: 6
smerPomerVyznamnychBodu: 2
smerPocetBoduProPrepocet: 1
smerPrvniPrepocet: 2

Ze 24 bodů je započítáno 12, výpočet je proveden 10-krát, vždy po dvou nalezených bodech. Výsledek bude velmi přesný, ale potrvá mnohem déle.



caraMaxPocetNalezenych: 6
smerPomerVyznamnychBodu: 1
smerPocetBoduProPrepocet: 1
smerPrvniPrepocet: 1

Ze 24 bodů je započítáno 21 výpočet je proveden 19-krát, po každém nalezeném bodu. Nejpřesnější a nejnáročnější řešení.



caraMaxPocetNalezenych: 6
smerPomerVyznamnychBodu: 3
smerPocetBoduProPrepocet: 2
smerPrvniPrepocet: 1

Ze 24 bodů je započítáno 9 výpočet je proveden 4-krát. První přepočet je proveden již po třetím nalezeném bodu. Ostatní po šesti.

Obr. 4.8: Příklady parametrů detekce nesvislých čar.

Výsledkem metody je množina všech nalezených čar. Dobrou vlastností algoritmu je, že nedetekuje svislé čáry, na které je určen první algoritmus. Vyplývá to ze způsobu hledání každého

dalšího bodu. Jak bylo uvedeno dříve, vždy se posuneme o jeden bod doprava, kde očekáváme pokračování čáry. Pokud je čára svislá, leží její body v jednom sloupci, proto nemohou být při průchodu zleva doprava souvisle označeny.

4.2 Vanishing point

V kapitole Návrh byly popsány dvě možnosti, jak hledat Vanishing point. První z nich je založena na náhodném výběru dvojice čar a počítání čar jim blízkých. Počet výběrů a tedy počet cyklů algoritmu volíme parametrem *pokusu*. Při testování algoritmu jsem volil hodnotu 60, což jsem považoval za dostatečné. Brzy jsem však zjistil, že se velmi často stává, že ty správné čáry pro určení Vanishing pointu nejsou vybrány ani jednou.

S vybranými čarami je třeba porovnat všechny ostatní z množiny nesvislých čar. Postupně budeme z množiny vybírat čáry, říkejme jim testovací, a počítat úhel, který svírají s čarami náhodně zvolenými. Parametr *maxUhelRovnobezky* určuje, jaký maximální úhel budeme tolerovat. Zajímají nás pouze ty testovací čáry, které jsou téměř rovnoběžné alespoň s jednou z vybraných čar. Parametr *maxUhelRovnobezky* jsem tedy nastavoval na hodnotu 5. Dále je potřeba spočítat vzdálenost testovací čáry od té z vybraných čar, se kterou je rovnoběžná. Parametrem *maxVzdalenost* nastavíme vhodnou hodnotu, třeba 5 bodů. Nakonec spočítáme všechny čáry, které splnily oba požadavky. Výsledná hodnota je hodnocením vybrané dvojice čar a Vanishing pointu, který definují svým průsečíkem. Výsledkem hledání je ten Vanishing point, který má největší hodnocení.

U takto nalezeného Vanishing pointu ale nelze tvrdit, že do něj směřuje nejvíce čar. V hodnocení byly počítány pouze čáry blízké těm, jejichž průsečík Vanishing point určil. Proto výsledek často nenaplňuje očekávání a rozhodl jsem se algoritmus pozměnit. V první řadě jsem vyloučil náhodnost a začal porovnávat vzájemně všechny čáry. A v druhé řadě jsem změnil systém hodnocení Vanishing pointu.

Druhá metoda začíná obdobně jako ta první. Vybírá dvojice čar z množiny všech nesvislých čar. Výběr je však systematický, prochází se všechny možné kombinace. Tyto dvě čáry svým průsečíkem urči Vanishing point. Jeho hodnocení je součtem hodnocení všech čar, které směřují do blízkosti Vanishing pointu. Tuto „blizkost“ určíme parametrem *maxVzdalenostOdVP*. Vzorec, podle kterého je hodnocení dané čáry spočítáno, je uveden a popsán v kapitole Návrh. Výsledky jsou ukládány do objektu *VanishingVysledky*, který poskytuje metodu pro získání nejlépe hodnoceného Vanishing pointu. V objektu jsou uchovávány všechny získané výsledky. Společně s nimi jsou uloženy i informace o všech čarách, které do daného Vanishing pointu směřují, a také individuální hodnocení každé této čáry. Tím jsou k dispozici informace pro kompletní vykreslení kterékoliv zpracované dvojice čar.

S těmito daty pracuje funkce *Vykresli*, která umí vykreslit jakoukoliv dvojici čar včetně Vanishing pointu, všech čar, které do něj směřují, a také hodnocení těchto čar. Díky tomu je možné podrobně analyzovat jednotlivé výsledky při nastavení různých parametrů. Příklad výstupu této metody je na Obr. 4.9. Vanishing point je mimo obraz. Tučné dlouhé čáry jsou přímky, které byly proloženy dvojicí čar určujících Vanishing point. U ostatních čar je uvedeno hodnocení spočítané podle vzorce uvedeného v kapitole Návrh. Většina těchto čar splývá s přímkami, důkazem jejich přítomnosti je číslo, udávající jejich hodnocení.

Obr. 4.9: Výstup metody pro vykreslení Vanishing pointu.

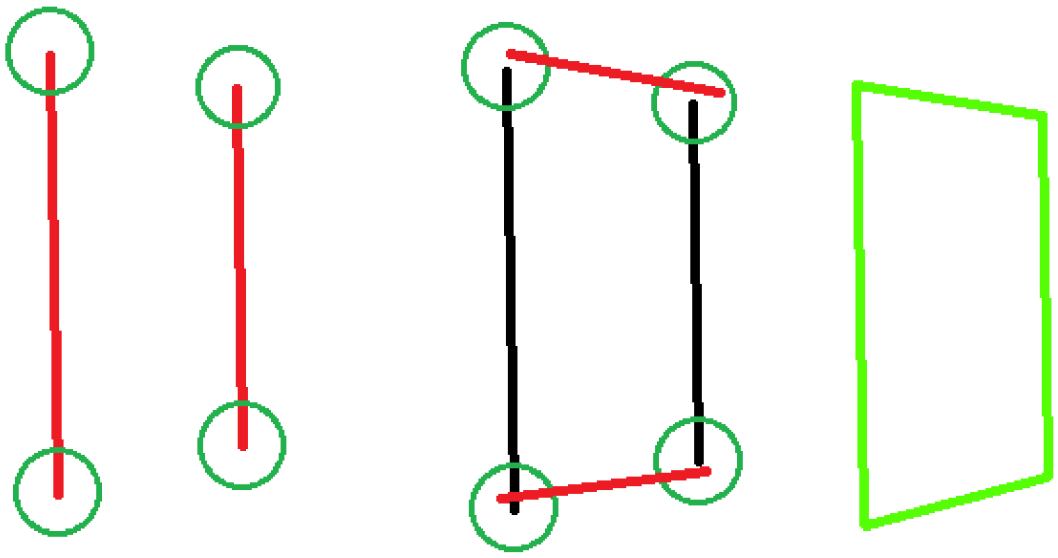
V programu jsou k dispozici obě metody pro získání Vanishing pointu. První metoda není příliš úspěšná, zato je mnohem méně výpočetně jednodušší. Druhá je robustní, ale náročná.

4.3 Označení dveří

Prvním krokem k označení dveří je volba vhodné dvojice svislých čar. Musí být zhruba stejně vysoké a vzdálenost mezi nimi musí být v patřičných mezích. Parametr *pomerVelikostiCar* určuje procentuální odchylku mezi výškou čar. Pokud nastavíme hodnotu 20, znamená to, že rozdíl výšek čar musí být menší, než 20 % délky té delší z nich. Vzdálenost čar nastavujeme parametry *minimalniVzdalenostPomer* a *maximalniVzdalenostPomer*. Z názvu vyplývá, že se jedná o poměr k délce čáry. Pokud nastavíme minimum na hodnotu 8 a maximum na hodnotu 3 znamená to, že vzdálenost mezi čarami musí být alespoň osmina délky první čáry a maximálně třetina délky první čáry. Z toho e patrné, že parametr *minimalniVzdalenostPomer* musí mít vyšší hodnotu, než parametr *maximalniVzdalenostPomer*.

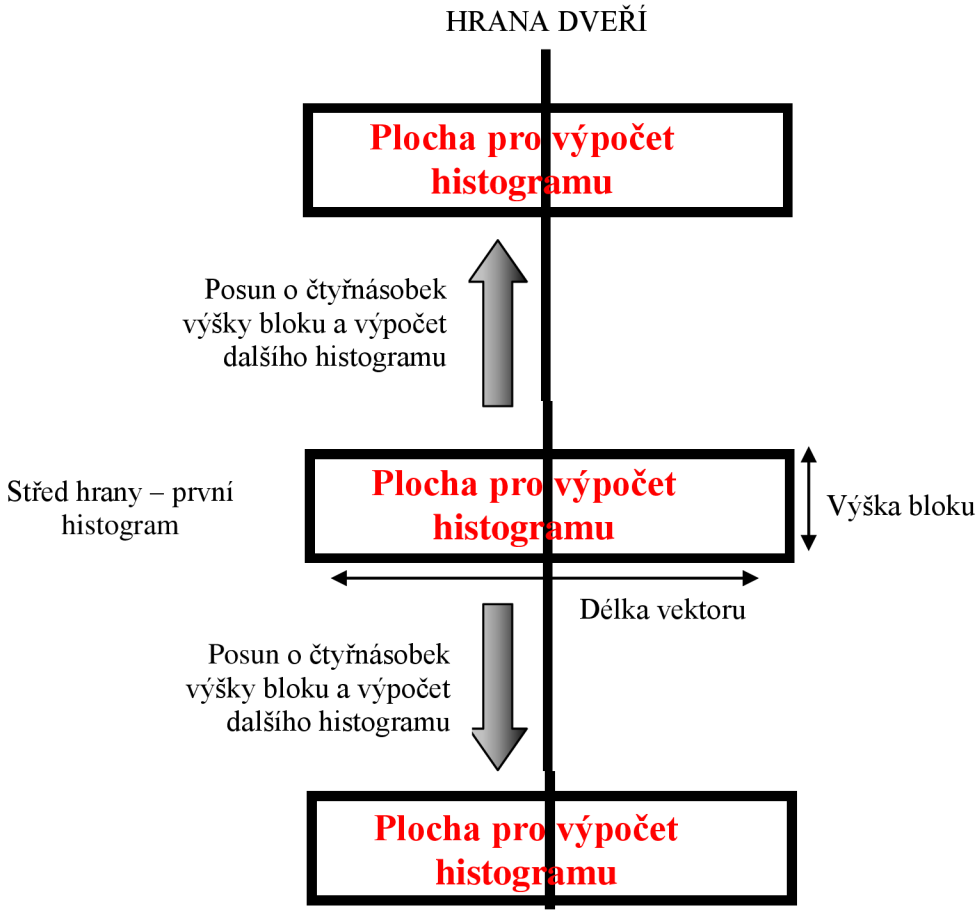
Všechny dvojice svislých čar, které splní výše uvedené, jsou uloženy do objektu *Dveře*. V něm je uchována informace o čtyřech bodech, které definují potenciální dveře. Zatím máme informaci pouze o bočních hranách. V dalším kroku tedy prohledáme všechny nesvislé hrany a zjistíme, zda některé z nich nemohou být horní nebo spodní hranou dveří, které jsme našli. Podle vzdálenosti hraničních bodu testovaných nesvislých čar od bodů uložených v objektu *Dveře* usoudíme, jestli daná nesvislá čára vyhovuje. Maximální vzdálenost nastavujeme parametrem *vzdalenostNesvislych*. Na Obr. 4.10 je červeně dvojice čar, která splnila podmínky výšky i

vzdálenosti. Zeleně označené body jsou uloženy v objektu *Dvere*. Poloměr kružnic zároveň znázorňuje maximální vzdálenost nesvislých čar od těchto bodů. Na prostředním náčrtku jsou červeně nesvislé čáry, které se nachází ve vymezené oblasti. Poslední náčrtek je výsledek detekce. Byly nalezeny všechny čtyři hrany. Do objektu *Dvere* je uložena informace o tom, které konkrétní hrany byly nalezeny.



Obr. 4.10: Hledání svislých a nesvislých čar a označení dveří.

Aby byla detekce přesnější, jsou označené dveře zkontrolovány metodou *KontrolaDveri*. Ta zjistí, zda horní a spodní hrana dveří směřuje do Vanishing pointu a na základě toho určí, jestli jsou nalezené dveře na zdi. Tato vlastnost je opět uložena v objektu společně s výsledkem. Posledním příznakem je změna jasového histogramu podél boční hrany dveří. Ilustrace je na Obr. 4.11. První histogramy jsou spočítány uprostřed levé a pravé hrany. Je možné nastavit šířku i výšku oblasti v obrázku, na které bude histogram počítán. Parametry se jmenují *delkaVektoru* a *vyskaBloku*. Aktuální histogram je uložen a posouváme se dále o čtyřnásobek výšky bloku. Na tomto místě spočítáme opět histogram. Sousední histogramy porovnáme korelační funkcí, kterou nabízí OpenCV. Výsledek porovnání uložíme do zásobníku a posuneme se opět dál. Pokud se dostaneme na okraj dveří, bude výsledek porovnání sousedních histogramů relativně nízký. Prahovou hodnotu můžeme nastavit parametrem *rozdil*. V blízkosti každého rohového bodu dveří by měl být detekován velký rozdíl v jasových histogramech. Pokud je nalezen alespoň u tří bodů, považujeme test rozdílného jasu za úspěšný.



Obr. 4.11: Počítání jasových histogramů.

Uživatel má tedy k dispozici tři příznaky, na jejichž základě může být rozhodnuto o tom, zda se o dveře jedná či ne. Jako výsledek detekce lze zobrazit čtyři obrázky:

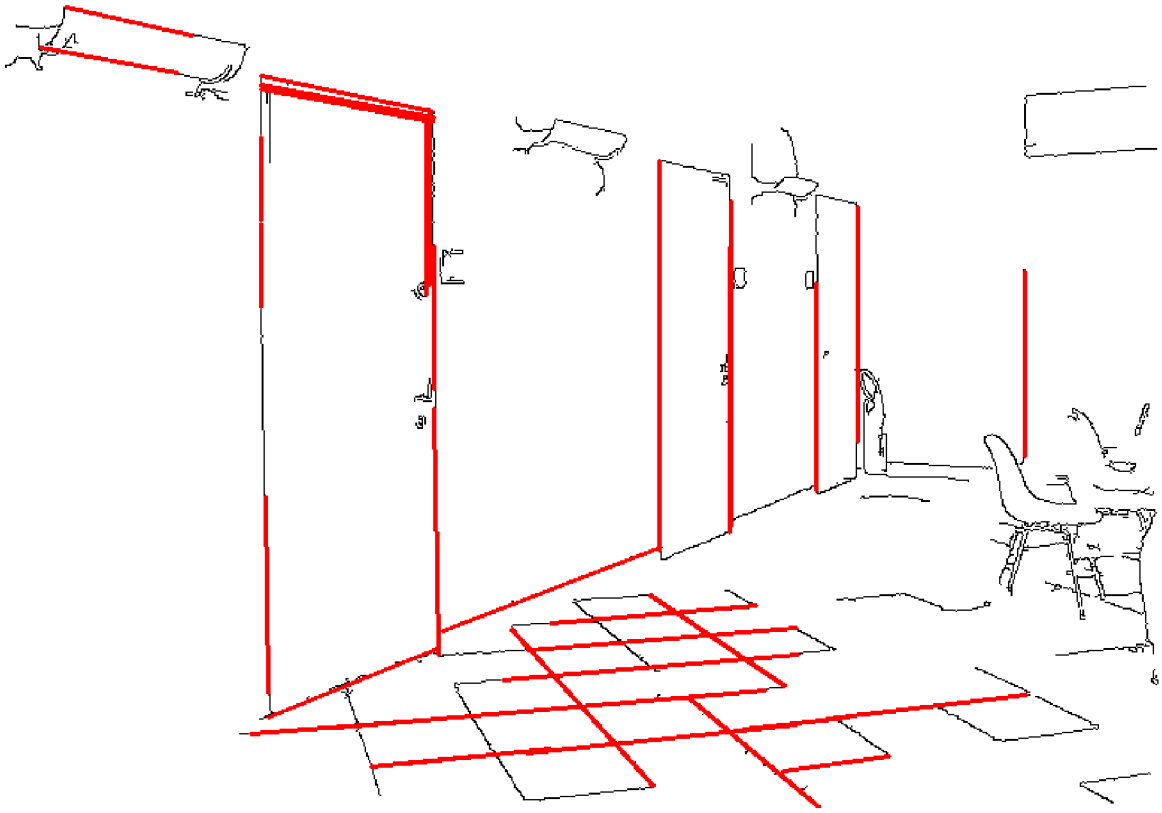
1. Kompletní dveře
2. Nekompletní dveře
3. Pravděpodobně dveře
4. Určitě dveře

Kompletní dveře jsou ty, u nichž byly nalezeny všechny čtyři hrany. Nekompletní mají buď tři, nebo dvě. Pravděpodobné dveře jsou takové, jejichž horní a spodní hrana směřuje do Vanishing pointu. Bez ohledu na kompletnost. Poslední označení je pro kompletní dveře, jejichž hrany směřují do Vanishing pointu a zároveň dveře uspěly v testu rozdílného jasu.

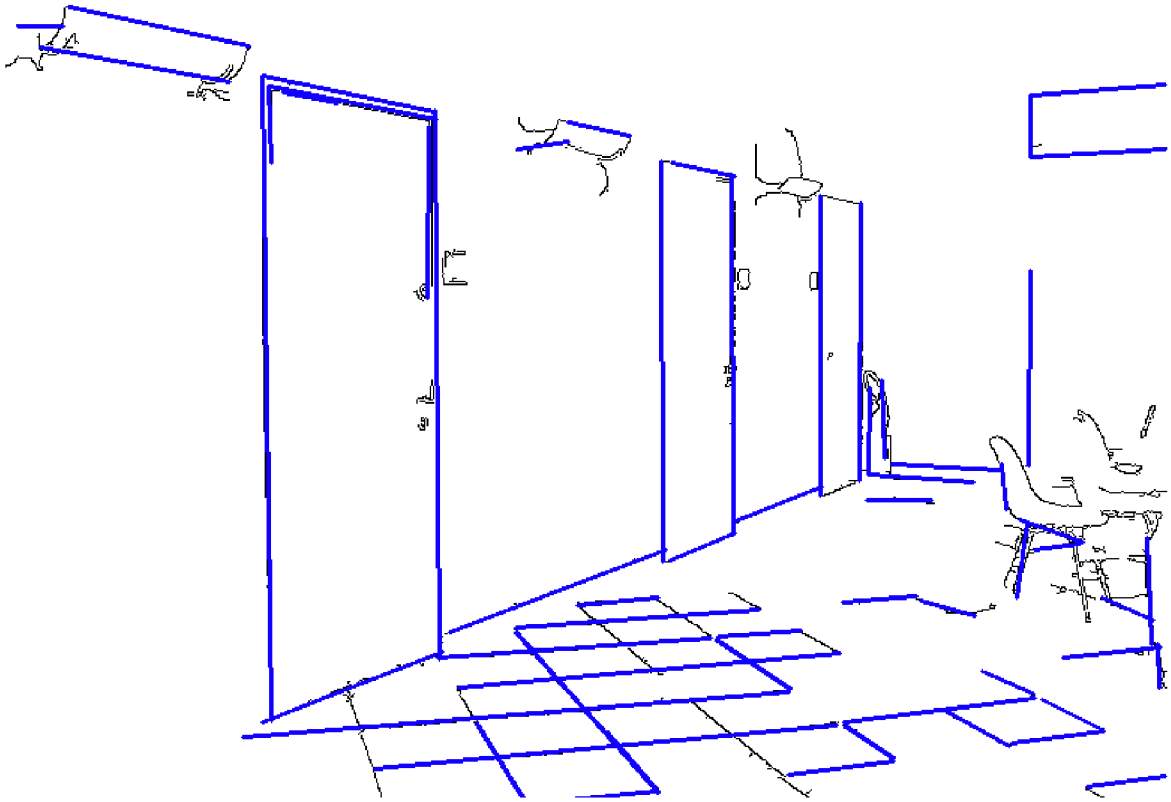
5 VÝSLEDKY

Systém pro detekci dveří funguje na většině snímků. Velkým problémem jsou rozmazané nebo příliš zašuměné fotografie, na kterých nefunguje Cannyho detektor a tedy ani další metody v pořadí. Další problém představuje dlažba na většině chodeb. Generuje velké množství čar, které znemožňují relevantní detekci Vanishing pointu.

Nejdříve srovnám výsledky Houghovy transformace a mého algoritmu pro detekci rovných čar. Z následujících obrázků je zřejmé, že můj algoritmus detekuje drtivou většinu čar. Houghova transformace má problém se spojitou detekcí svislých čar a celkově s detekcí čar nesvislých. Viz druhé a třetí dveře. Černé čáry na obrázcích jsou výstupem Cannyho detektoru.

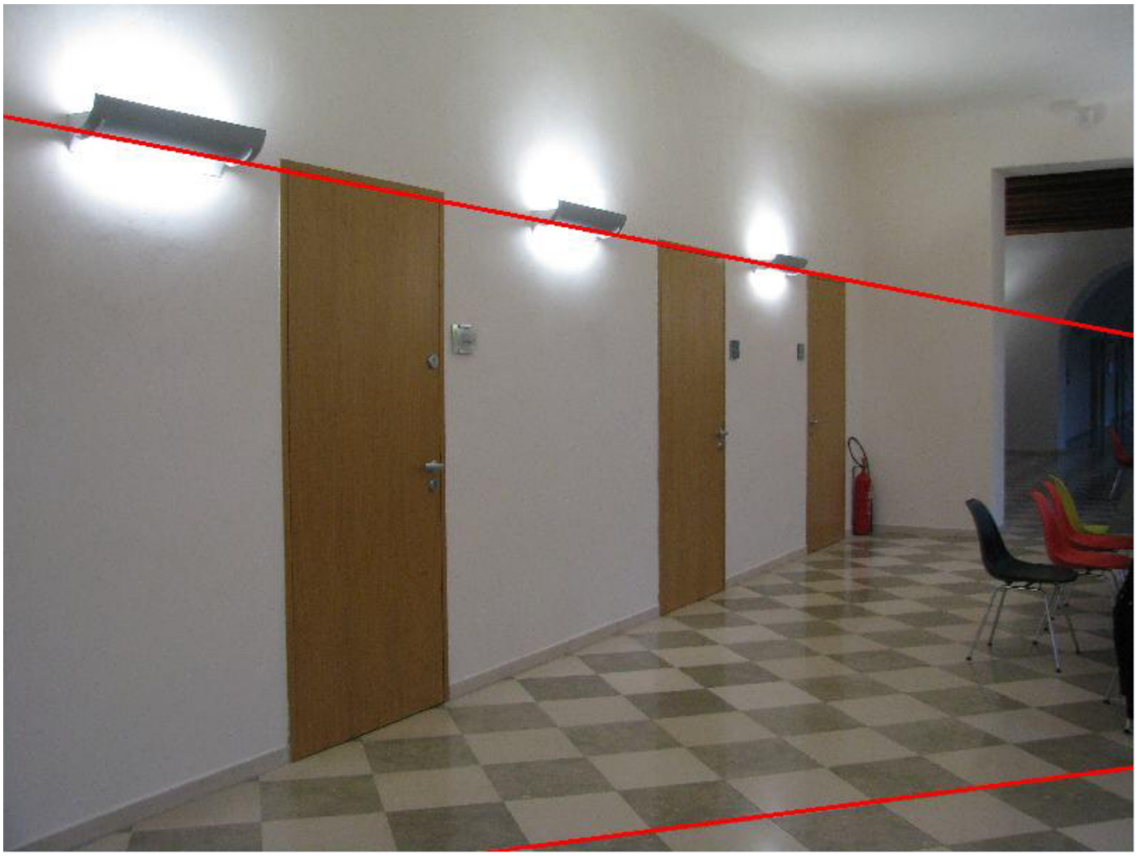


Obr. 5.1: Houghova transformace, na pozadí Canny.

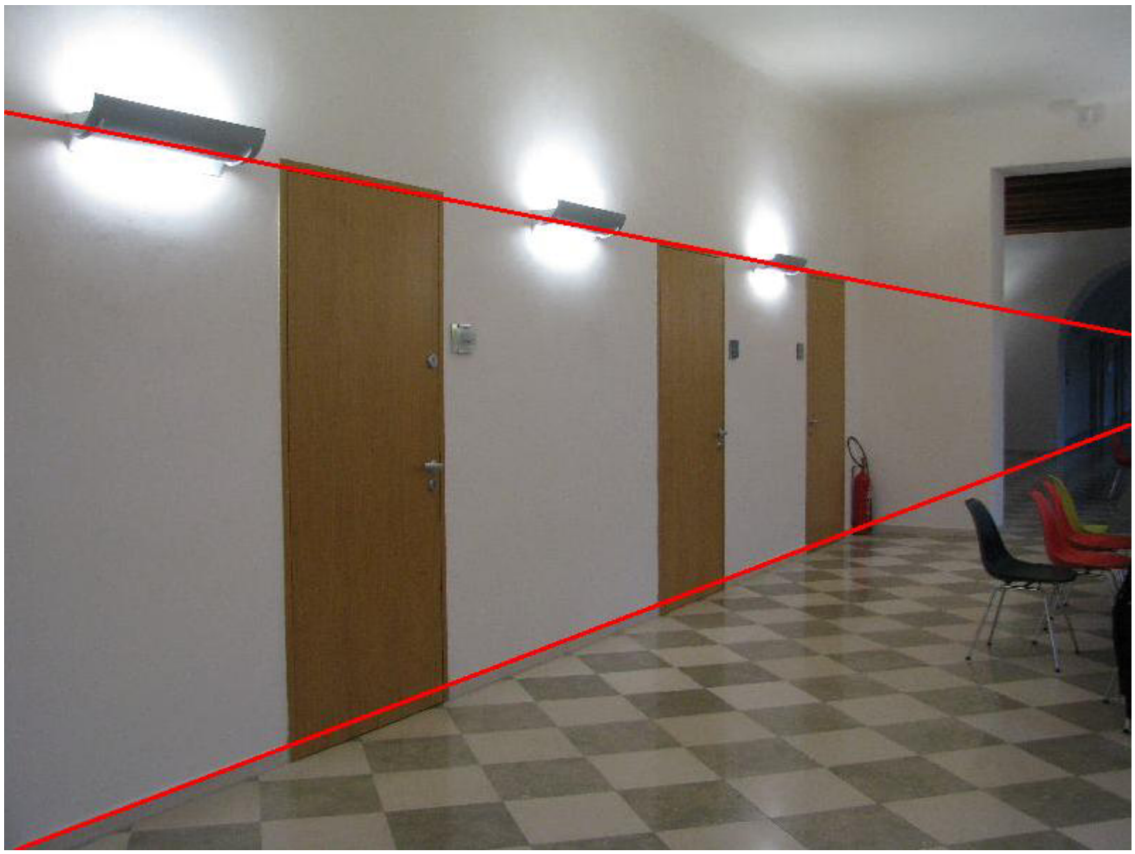


Obr. 5.2: Vlastní algoritmus detekce čar, na pozadí Canny.

Další obrázky nabízí srovnání dvou metod hledání Vanishing pointu. První z nich vybírá dvojice čar náhodně a výsledek je hodnocen počtem čar, které jsou blízko těm vybraným. Druhý algoritmus vybírá systematicky všechny možné dvojice a hodnocení výsledku závisí na počtu, délce a blízkosti čar, které směřují do Vanishing pointu. Tento algoritmus podává na jednom obrázku vždy stejné výsledky, protože není založen na náhodném výběru, a celkově je mnohem úspěšnější, než ten předchozí. Velké úspěšnosti napomáhá hlavně kontrola polohy nalezeného Vanishing pointu. Dříve se velmi často stávalo, že velké množství dlaždic ovlivnilo výpočet mnohem více, než skutečné hrany dveří. Výsledkem byl průsečík někde na podlaze. Díky podmínce minimální výšky Vanishing pointu není takový výsledek brán v úvahu a hledá se další.

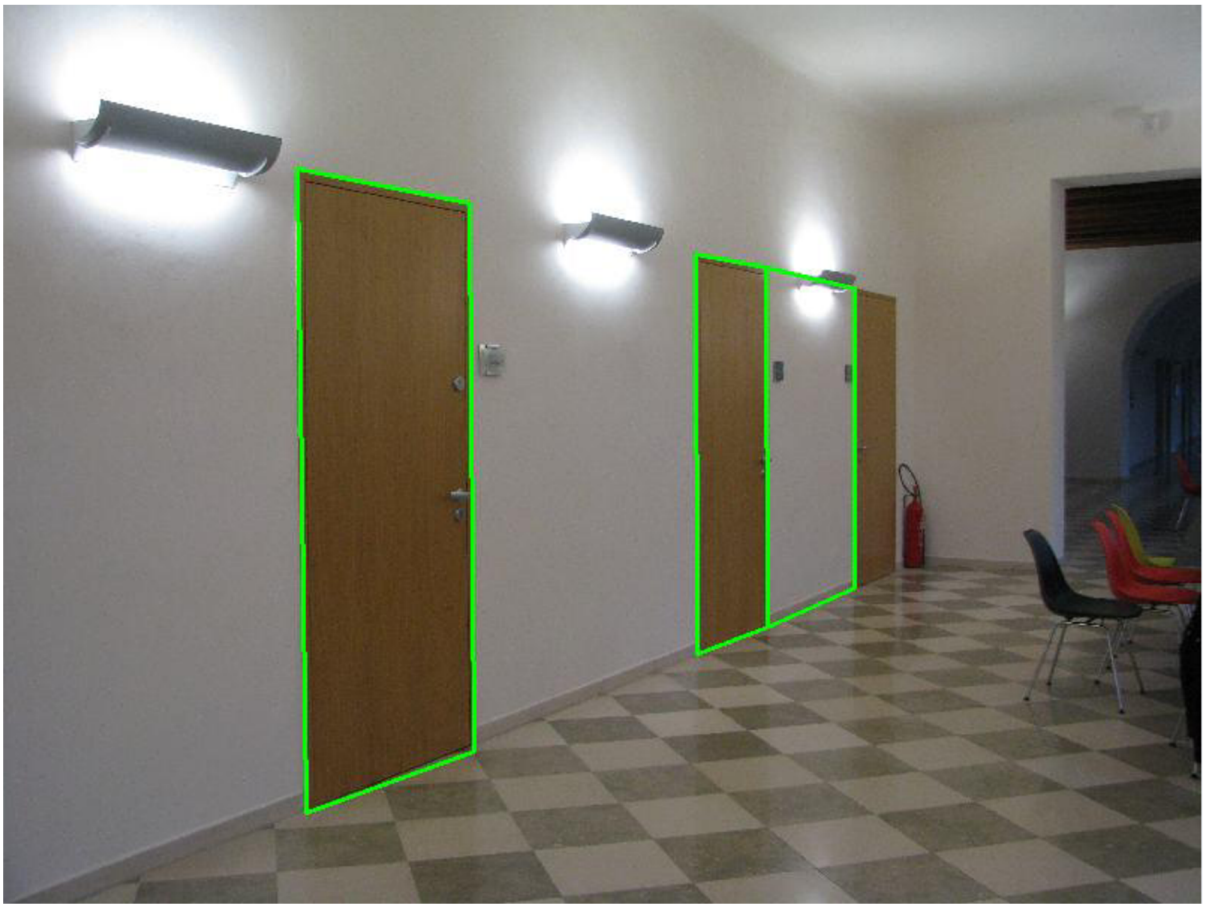


Obr. 5.3: Algoritmus pro detekci Vanishing pointu s náhodným výběrem čar.



Obr. 5.4: Algoritmus pro detekci Vanishing pointu s výběrem všech čar.

Poslední srovnání se týká konečného vyhodnocení. Uživatel si může vybrat mezi dvěma různými zobrazeními výsledku: *Pravděpodobně dveře* a *Určitě dveře*. První možnost je benevolentnější a označí i dveře, které úplně nevyhověly všem požadavkům. Výsledek ale nelze považovat za stoprocentní. Někdy se stává, že jsou označena i místa, která ve skutečnosti nejsou dveřmi. Tato situace se objevila na následujících obrázcích. Pokud však chceme nalézt co největší počet existujících dveří, doporučuje se používat právě tento způsob reprezentace výsledků.



Obr. 5.5: Pravděpodobně dveře - nepřesné, vhodné i pro méně kvalitní snímky.

Druhý způsob reprezentace (*Určitě dveře*) označí pouze ty dveře, které prošly v pořádku všemi testy. V praxi se ale ukázalo, že je takto mnoho dveří opomenuto. Dveře jsou detekovány pouze na kvalitních a bezproblémových snímcích. Výhodou však je, že se téměř nestává, aby bylo označeno místo, kde dveře ve skutečnosti nejsou. Viz Obr. 5.6.

Obr. 5.6: Určitě dveře – přesná detekce.

Na Obr. 5.6 lze také pozorovat, že třetí dveře nejsou detekovány. Velký šum ve fotografiích zapříčiňuje, že u vzdálenějších dveří selhává Cannyho detektor hran. Není tedy možné provést rozpoznání takových dveří.

Program pracuje pouze se statickými obrázky. Pro vylepšení úspěšnosti detekce by bylo velmi přínosné zpracovávat video. Obecně není tolik zatíženo šumem a nabízí mnohem více pohledů na totéž místo. Při tomto rozšíření by bylo možné využít bez výjimky všechny stávající metody, které se podílí na detekci. Zároveň se nabízí několik dalších možností, které není možné provádět u statického obrazu. Bylo by možné sledovat význačné čáry v obraze a z nich počítat Vanishing point snadněji než nyní. Také by odpadl problém s nemožností detekce částečně překrytých dveří, díky více úhlům pohled na totéž místo.

Proti šumu ve fotografiích a jím způsobené chybné detekci hran by mohly pomoci další klasifikátory, které by se zaměřily na odlišné parametry snímku, než jsou hrany. Pomoci by mohla metoda segmentování obrazu na základě textury. Podle ní mohly být znovu detailně prohledány ty části obrazu, které mají podobnou texturu jako dříve nalezené dveře. Dalšími vodítky by mohly být malé příznaky, které se často u dveří vyskytují. Například klika, mezera pode dveřmi a tak podobně.

Přesto, že současný stav systému umožňuje případné nasazení v praxi, mohla by být jeho použitelnost znásobena přidáním nových modulů pro detekci jiných objektů, než dveře. Například okna, svítidla nebo rostliny.

6 ZÁVĚR

Úspěšně byl navržen a implementován systém, který umožňuje detekci dveří. Samotnému označení dveří předchází lokalizace zdi, podlahy a stropu. Díky tomu lze snadno přidat funkcionalitu pro detekci dalších objektů (okna, svítidla, podlahová krytina). Na kvalitních snímcích jsou dveře s velkou pravděpodobností nalezeny. Mylná detekce, respektive označení místa, kde se ve skutečnosti žádné dveře nevyskytují, se objevuje pouze zřídka. Proces detekce je navržen tak, aby se dal snadno parametrizovat a přizpůsobit nestandardním lokacím či snímkům s vyšším rozlišením.

Přínos dosavadní práce vidím hlavně v kvalitní přípravě dat pro provádění detekce objektů v obraze. Jedná se především o vlastní algoritmus pro rozpoznávání rovných čar a robustní systém pro detekci Vanishing pointu včetně metody pro vykreslení všech nalezených kandidátů a jejich hodnocení (vhodné pro testování a případné rozšíření).

Systém detekce by mohl být později rozšířen o zpracování videa místo statických obrázků. Přineslo by to spoustu nových možností, jak z obrazu získat informace. Pro spolehlivější označování dveří by bylo vhodné do budoucna vytvořit metody pro zpracování dalších klasifikátorů. Například rozčlenění snímku na části podle textury nebo hledání kliky na dveřích či mezery pode dveřmi. Díky tomu by se zvýšila pravděpodobnost správného označení dveří na snímcích s nižší kvalitou, kde selhává detektor hran.

Literatura

[1] CHUM, Ondřej. *Two-View Geometry Estimation by Random Sample and Consensus*. 2005.

[2] CHEN, Z; BIRCHFIELD, S. T. *Visual detection of lintel-occluded doors from a Single image*. IEEE Computer Society Workshop on Visual Localization for Mobile Platforms. 2008, s. 1-8.

[3] HENSLER, Jens; BLAICH, Michael; BITTEL, Oliver. *Real-time Door Detection Based on AdaBoost Learning Algorithm*. International Conference on Research and Education in Robotics [online]. 2009. Dostupný z WWW: <http://www-home.fh-konstanz.de/~bittel/Publikationen/Publikationen.htm>.

[4] CANNY, J. *A Computational Approach To Edge Detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–714, 1986.

[5] BRADSKI, Gary; Kaehler, Adrian. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 1st edition, September 2008.

Seznam příloh

Příloha 1. Příklady detekce dveří

použity výsledky pravděpodobných dveří – někdy nepřesné

Příloha 1: Příklady detekce dveří

Příloha 1: Příklady detekce dveří