Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

Diplomová práca

Marek Mateják

Lokalizácia robota pomocou 3D mapy

Katedra softwarového inžinírství
Vedoucí diplomové práce: Mgr. Zbyněk Winkler
Studijní program: Počítačová grafika

Chcel by som sa poďakovať hlavne ľuďom, ktorý ma počas môjho štúdia a písania diplomovej práce podporovali. Ďalej priateľom, s ktorými som sa podieľal na vytvorení robota do súťaže Eurobot 2006. Vedúcemu diplomovej práce za podporu a zapožičanie kamery. A všetkým, s ktorými som jednotlivé elementárne problémy práce konzultoval.

Prehlasujem, že som svoju diplomovú prácu napísal samostatne a výhradne s použitím citovaných prameňov.

Súhlasím so zapožičiavaním práce.

V Prahe dňa 10.8.2006

Obsah

Název práce:	Lokalizácia robota pomocou 3D mapy
Autor:	Marek Mateják
Katedra (ústav):	Katedra softwarového inžinírství
Vedoucí diplomové práce:	Mgr. Zbyněk Winkler
e-mail vedoucího:	Zbynek.Winkler@mff.cuni.cz

Táto práca pojednáva o možnostiach lokalizácie autonómneho mobilného robota pomocou kamery v známom prostredí. Podrobne je rozobraný model zobrazenia scény perspektívnou projekciou. Aj výpočet polohy na základe spárovaných bodov zo scénou. Špeciálne pre model robota s pevným umiestnením kamery, ktorý sa pohybuje po rovnej podlahe. Lokalizácie je riešená z dvoch hľadísk. Z globálneho, kde je potreba určiť pozíciu robota v mape bez predchádzajúcej znalosti jeho pozície. A z lokálneho hľadiska, kde je pozícia robota sledovaná počas jeho pohybu scénou. V prvom prípade sa odhad polohy opiera hlavne na farebnom rozlíšení segmentov. Analyzovaný je prípad nerozlíšiteľných oporných bodov, kde je poloha odhadovaná na základe ich priestorového rozloženia. Na druhej strane pri známej predchádzajúcej pozícii krátkeho pohybu robota sú k dispozícii dve metódy sledovania pohybu oporných bodov na snímku z kamery. A to známy algoritmus ICP na registráciu bodov na snímku a algoritmus Mean Shift na sledovanie výrazných oblastí obrázku. Na záver je uvedený návrh Monte Carlo filtrovania, ktoré zaisťuje robustnosť pri sledovaní segmentov. Klíčová slova: lokalizace robota, model kamery, perspekívna projekce, trackování polohy

Title:	Robot localization by 3D map
Author:	Marek Mateják
Department:	Department of Software Engeneering
Supervisor:	Mgr. Zbyněk Winkler
Supervisor's e-mail address:	Zbynek.Winkler@mff.cuni.cz

This thesis deals about possibilities of autonomous mobile robot localization by camera in the known environment. In detail the model is analyzed of scene representation by perspective projection including evaluation of the location according to paired points with the scene. This is done especially for the model of the robot with the fixed camera location that moves on the flat floor. Localization is analyzed from two points of view. From the global view, where is the goal to evaluate position without previous knowledge of the position. And from the local view, where is the location of the robot tracked during its movement in the scene. In the first case is estimation of the position based mainly on the color separation of segments. The case of undistinguishable reference points is analyzed, where the position is estimated according to their field location. On the other hand, when the previous position is known, the two methods of reference points tracking are mentioned. First method is ICP algorithm, which makes the registration of points. The second is the Mean Shift algorithm for tracking image segments. In the end is mentioned the application of Monte Carlo filtration, which assures robustness of segment tracking. Keywords: robot localization, camera model, perspective projection, tracking

Kapitola 1

Problém lokalizácie pomocou 3D mapy

1.1 Úvod

Práca rieši problém ako lokalizovať mobilného robota, ktorého jediným senzorom je kamera. Lokálizácia pritom prebieha vo vopred známom prostredí. Robot sa môže po tomto prostredí pohybovať iba po rovnej podlahe. Pozíciou robota sa myslí pozícia ohniska kamery, takže robot by mal vedieť aspoň relatívnu polohu kamery vzhľadom k jeho polohe a orientácii.

Keďže jedinou informáciu o prostredí je obraz z kamery, tak je nutné vedieť vyjadriť perspektívnu projekciu zobrazovania scény pomocou zrovnania obrazu s mapou. A nielen to, je potrebné zistiť ako z danej projekcie vyjadriť natočenie a pozíciu robota v scéne.

Problém takto definovanej lokalizácie mobilného robota v známom prostredí sa rozdeľuje na dve úlohy. Prvou je určiť polohu robota v mape ak o svojej pozície nemá žiadne informácie. A inou úlohou je sledovať jeho pozíciu počas pohybu robota.

Hľadanie pozície, ak sa robot môže vyskytovať kdekoľvek v mape, je z princípu problém nájdenia oporných bodov v obraze z kamery. Problémom však je ak sú tieto body na rôznych miestach rovnaké. Vtedy je možné zohľadniť rozloženie rôznych typov viditeľných bodov a vyberať miesto, kde takéto rozloženie najviac vyhovuje zaznamenanému okoliu robota. Oporné body je tak treba rozlíšiť natoľko, aby bolo takýchto situácií, čo najmenej. Na rozlíšenie oporných bodov v obrázku z kamery je možné použiť akékoľvek

príznaky, ktoré sú invariantné na perspektívnu projekciu.

Pri pohybe robota scénou sa sleduje pohyb zvolených bodov na obrázku. Keďže je známa pozícia v predchádzajúcom kroku, tak je možné tieto body voliť pomocou mapy tak, aby bolo možné ich sledovanie na obrázku.

Nie každá scéna však umožňuje takúto lokalizáciu robota pri ľubovoľnom možnom pohľade kamery do nej. Pokiaľ nie je možné na snímku kamery identifikovať žiadne oporné body, ani body na sledovanie pozície, tak bohužiaľ takýto postup zlyháva. Nič však nebráni tomu, aby boli do priestoru na takýchto miestach pridané vhodné body externe a tým zaručená lokalizácia v celom prostredí. Ďalšou možnosťou je použiť širokouhlý objektív, ktorého perspektívna projekcia sa líši iba v uhle záberu. Tak sa stávajú viditeľné aj body, ktoré sú od seba vzdialenejšie.

Kapitola 2

Prepočet súradníc

Ukazuje sa, že ideálny prípad snímku pre zrovnávanie so scénou je snímok zriadený kamerou, ktorá nemá žiadne skreslenie a je orientovaná kolmo nadol (resp. nahor). Do takéhoto modelu je možné previesť ľubovoľne orientovanú kameru. Prepočtu je venovaná celá nasledujúca kapitola. V sekcii 2.1 popisuje jednoduchý model skreslenia kamery, ktorý používa napríklad knižnica OpenCV [4]. V sekcii 2.2 o priestore kamery sa popisuje ideálna perspektívna projekcia. A to, ako snímok z kamery pozerajúcej šikmo na podlahu natočiť do ideálneho modelu, je rozpísané v sekcii 2.3 o priestore robota. Rovina ideálneho snímku je nazvaná referenčnou rovinou (sekcia 2.4). Tá si ako výsledok nášho skúmania prepočtu súradníc zaslúži značnú pozornosť, pretože ďalšie výpočty pozície robota v kapitolách 3 a 5 sú založené prevažne nad ňou. Nakoniec je pridaná úvaha o dvoch skalibrovaných kamerách, ktoré pridávajú hĺbkovú informáciu k detekovaným bodom.

2.1 Kalibrácia kamery

Úlohou kalibrácie kamery je určiť interné parametre kamery tak, aby bolo možné obrázok čo najjednoduchšie mapovať do scény. V tomto kroku je vhodné započítať taktiež deformáciu šošovkou kamery. Parametre týchto výpočtov sú zviazané s kamerou. Spočítajú sa iba raz a zostávajú konštantné.

Upravenie súradníc bodov na obrázku z indexov pixelov ide v homogénnych súradniciach spočítať pomocou matice K. Polohy bodov sa upravia zmenou mierky v x-ovom smere násobením s f_x a podobne v y-ovom smere

podľa f_y. A počiatok sa presunie tak, aby určoval smer pohľadu. Mierka zabezpečuje, že výsledný bod (u, v) reprezentuje priamku prechádzajúcu počiatkom (ohniskom kamery) s orientáciou $(u, v, 1)$.

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}; \quad K^{-1} = \begin{pmatrix} \frac{1}{f_x} & 0 & -\frac{c_x}{f_x} \\ 0 & \frac{1}{f_y} & -\frac{c_y}{f_y} \\ 0 & 0 & 1 \end{pmatrix}; \quad \begin{pmatrix} a \\ b \\ 1 \end{pmatrix} = K \times \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{2.1}$$

Súradnice pixelu na obrázku sú (X, Y). Skreslenie šošovkou kamery je možné zapísať napríklad rovnicami:

$$u = a \cdot (1 + k_1 r^2 + k_2 r^4) + 2p_1 ab + p_2(r^2 + 2a^2)$$
$$v = b \cdot (1 + k_1 r^2 + k_2 r^4) + 2p_2 ab + p_1(r^2 + 2b^2)$$

kde

$$r^2 = a^2 + b^2$$

k_1, k_2 sú polárne deformačné koeficienty

p_1, p_2 sú tangenciálne deformačné koeficienty

Existuje mnoho metód a algoritmov na skalibrovanie interných parametrov kamery. Detaily kalibrácie sú nad rámec tejto práce. Pre naše účely bude postačovať uvedený model, ktorý je napríklad výsledkom kalibrácie kamery nad šachovnicovou podlahou pomocou knižnice OpenCV [4].

2.2 Priestor kamery

Natiahnutý a posunutý obrázok z kamery ako výsledok kalibrácie je vhodné pred napasovaním do scény jemne upraviť. Najprv sa nastaví os y tak, aby smerovala nahor, pretože pre väčšinu kamier sú súradnice snímkov orientované po riadkoch - zhora nadol. A potom nastane korekcia otočenia okolo osi pohľadu tak, aby os x bola vo výsledku rovnobežná s podlahou. Pokiaľ je kamera orientovaná kolmo nadol, tak by os y na obrázku mala odpovedať orientácii robota, tj. smer dopredu. Pri otáčaní figuruje uhol β, ktorý označuje toto vychýlenie súradníc bodov v protismere hodinových ručičiek. Naviac je tento priestor rozšírený o os z, ktorá je orientovaná v smere pohľadu tak, aby bola perspektívna projekcia do tejto roviny obrázku v hĺbke $z = 1$ realizovateľná iba vydelením podľa z-ovej súradnice. Možno si to predstaviť ako rozšírenie skalibrovaného obrázku do homogénnych súradníc. V tejto sústave sú všetky body, ktoré sa premietajú do rovnakého bodu roviny $z = 1$ nerozlíšiteľné. V praxi to znamená,

že pokiaľ nemáme ďalšie informácie, tak nevieme, ako je pozorovaný bod na obrázku z kamery ďaleko.

Otočenie okolo osi pohľadu je vhodné skalibrovať manuálne zrovnaním kamery tak, aby pri kolmom pohľade na horizontálne čiary zostali horizontálne aj na snímkoch. Po takejto kalibrácii netreba uvažovať započítavanie matice Q_3.

$$A_{cam} = (x, y, z)^T = Q_3 \times C_y \times (\lambda(u, v, 1)^T); \quad A_{img} = (u, v)^T \tag{2.2}$$

$$Q_3 = \begin{pmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad C_y = C_y^{-1} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Parameter λ tu predstavuje hĺbku bodu v tomto priestore. Priestor kamery má teda počiatok v ohnisku kamery. Os z určuje smer pohľadu a os y smeruje nahor po rovine snímku. Sústava tohoto priestoru je ľavotočivá. Odpovedá to pravidlu ľavej ruky, keď sa uchopí os z tak, aby palec určoval jej smer a prsty určovali kladný smer otáčania (tj. v protismere hodinových ručičiek) na rovine x, y. V tomto smere sa taktiež natáčajú korektnejšie súradnice do súradníc bodov z kalibrácie kamery o uhol β. Ak teda otočím v tomto smere s bodom z kalibrácie A_{img}, tak získavam jeho korigované súradnice v priestore kamery. Dá sa to predstaviť aj ako situácia, keď bod zostáva na mieste a priestor sa natáča okolo osi z o uhol $-\beta$ oproti nemu. V ďalšom texte je často používaný pojem priestor snímku, ktorý označuje taktiež priestor kamery.

2.3 Priestor robota

Pre výpočet zobrazenia medzi scénou a obrázkom z kamery je nutné poznať natočenie kamery a pozíciu ohniska kamery. Tieto informácie sú však výstupom problému lokalizácie, ktorý má určiť, kde sa nachádza robot. Jeho pozícia je určená pozíciou na podlahe a horizontálnym natočením kamery (jeho orientáciou). Takže je vhodné prevod medzi scénou a obrázkom doplniť o akýsi medzikrok, ktorý ešte nie je závislý na pozícii robota. Predpokladom však je, že pri tomto výpočte bude známy uhol δ vychýlenia kamery od pohľadu kolmo nadol. Týmto predpokladom sa veľa nestratí, pretože tieto parametre sú úzko spojené s konštrukciou robota. Pokiaľ by robot mohol kamerou hýbať, tak je dobré, aby mal o tom informácie a vedel novú relatívnu polohu kamery vzhľadom k jeho súradniciam. Ďalej však

vo väčšine prípadov túto možnosť nepripúšťam a zaoberám sa iba zjednodušeným modelom, keď má kamera konštantnú výšku aj sklon k podlahe.

Priestor robota sa teda líši od priestoru kamery natočením a orientáciou osi z tak, aby osi x a y ležali na rovine rovnobežnej s podlahou. Takéto natočenie je možné realizovať okolo osi x. Výhodou je, že pokiaľ kamera smerovala dopredu, tak orientácia robota zostáva určená osou y.

Keďže je os z v priestore kamery orientovaná opačne (smerom nadol, pri pohľade z vrchu na snímok) než v priestore scény (smerom nahor, určuje výšku bodov nad podlahou), tak je potrebé po natočení snímku do roviny rovnobežnej s podlahou ešte zmeniť orientáciu osi z.

$$A_{rbt} = C_z \times (Q_2 \times (\lambda A_{cam})); \qquad \lambda A_{cam} = Q_2^{-1} \times (C_z \times A_{rbt}) \tag{2.3}$$

$$Q_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\delta & -\sin\delta \\ 0 & \sin\delta & \cos\delta \end{pmatrix}, \quad Q_2^{-1} = Q_2^T, \quad C_z = C_z^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

Priestor robota má teda počiatok v ohnisku kamery, osi x, y sú rovnobežné s podlahou a os z smeruje nahor. Tento priestor už nadobudol zaužívanejšiu pravotočivú formu, kde pri uchopení osi z (palec smeruje od počiatku) pravou rukou, určujú prsty kladný smer otáčania po rovine xy.

2.4 Referenčná rovina pohlahy

Referenčnou rovinou budem nazývať rovinu rovnobežnú s podlahou vzdialenú 1 od ohniska kamery smerom k podlahe. Sú to normované ($z = 1$) body v priestore robota v závislosti na ich výške nad podlahou.

Zo vzťahov pre body v relatívnom priestore robota je možné zrátať jednoznačný prevod medzi rovinou snímku v priestore kamery a referenčnou rovinou.

$$A_{cam} = (i, j, 1)^T = (\frac{r}{-s\sin\delta - \cos\delta}, \frac{s\cos\delta - \sin\delta}{-s\sin\delta - \cos\delta}, 1)^T \tag{2.4}$$

$$A_{ref} = (r, s, 1)^T = (\frac{i}{-j\sin\delta - \cos\delta}, \frac{j\cos\delta - \sin\delta}{-j\sin\delta - \cos\delta}, 1)^T \tag{2.5}$$

Tu treba poznamenať, že do referenčnej roviny nie je možné projektovať body, ktoré ležia na horizonte podlahy, alebo sú vo výške kamery (tj.

$j = -1/\tan(\delta)$, resp. - $-j\sin\delta - \cos\delta = 0$). Dokonca projekcia im blízkych bodov je už hodne nepresná, viď. sekciu 3.2. Pre účely lokalizácie robota cez referenčnú rovinu budú teda zaujímavé body, ktoré sú v blízkosti robota dostatočne pod úrovňou kamery. Symetricky je však možné pohľad rozdeliť do časti nad úrovňou kamery a pod ňou. Algoritmus potom aplikovať paralelne na obidve referenčné roviny. Ideálny model tvorí robot, ktorého kamera je umiestnená dostatočne vysoko aby nemusela zaberať body, ktoré ležia v jej výške. Poprípade jej obraz rozdeliť alebo orezať.

Referenčná rovina robota je zaujímavá najmä tým, že je možné do nej ľahko previesť body z obrázku. Ďalej, že každá lineárna transformácia bodov v tejto rovine odpovedá rovnakej lineárnej transformácii v každej rovine rovnobežnej s podlahou. A posun bodov v určitej výške nad podlahou odpovedá v nej posunu násobeným s koeficientom γ. Koeficient γ pritom nie je nič iné, ako rozdiel výšky kamery a výšky bodu nad podlahou. Výška kamery rovnako ako sklon k podlahe patrí taktiež k interným parametrom robota. Je teda možné robota uspôsobiť tak, aby boli tieto parametre konštantné, alebo aby vedel ich aktuálne hodnoty. Poprípade stačí, keď robot vie zistiť všeobecné vyjadrenie koeficientov perspektívnej projekcie. V tomto prípade sa spúšťa automatická kalibrácia relatívnej polohy kamery vzhľadom k polohe robota. Táto kalibrácia spočíva v zrátaní posunu a 3D rotačnej matice projekcie (napríklad pomocou algoritmov v sekcii 3.3 resp. 3.4 predtým je nutné spárovať body bez použitia referenčnej roviny napríklad pomocou klasifikácie podľa histogramu ako je uvedené v sekcii 4.4). Vyjadrením matice rotácie a posunu pre perspektívnu projekciu je tak možné vyextrahovať interné parametre robota (sklon kamery k podlahe, natočenie obrazu okolo osi pohľadu, alebo výšku kamery nad podlahou) podľa vzťahov zo sekcie 2.6. Inou metódou ako nájsť prevod do referenčnej roviny je parametrizovanie priamo zobrazenia medzi snímkom kamery a referenčnou rovinou. Získa sa tak opäť iba matica všeobecného natočenia v 3D, ktorú je možné vyjadriť tiež ako $Q_2 \times Q_3 \times C_y$, resp. jej inverzný tvar. Keďže tieto matice otočenia sú všetky ortonormálne, tak inverzná matica je rovná matici transponovanej. Rôzne metódy hľadania koeficientov priamo tejto matice sú uvedené napríklad v S.Baker,2006 [7].

Veta 1 *Bod v priestore robota projektovaný do referenčnej roviny je rovnaký ako bod projektovaný do snímku v súradniciach priestoru kamery a pomocou vzťahu 2.5 z neho nakoniec vyjadrený bod v referenčnej rovine.*

Dôkaz: Nech $(x, y, z)^T$ je bod scény vyjadrený v relatívnom priestore robota. To znamená, že jeho súradnica z je rozdielom výšky kamery od výšky

bodu v scéne nad podlahou. Potom jeho projekcia do referenčnej roviny je $(x/z, y/z, 1)^T$. Jeho vyjadrenie v priestore kamery je $\lambda \cdot Q_2^{-1} \times C_z \times (x, y, z)^T$. V rovine snímku je parameter λ vyjadrený tak, aby bola posledná súradnica hĺbky rovná jednej ($\lambda = 1/(-y \sin\delta - z\cos\delta)$). Ďalej sa tento bod snímku ($(i, j, 1) = (1/(-y\sin\delta - z\cos\delta)) \cdot Q_2^{-1} \times C_z \times (x, y, z)^T$) dosadí do vzťahu 2.5 a tak zobrazí v referenčnej rovine. A nakoniec vidíme, že naozaj dostávame bod $(x/z, y/z, 1)^T$.

Veta 2 *Na výpočet pozície nad referenčnou rovinou sú postačujúce dva rozdielne páry bodov snímku a mapy.*

Dôkaz: Nech sú (X_0, Y_0), (X_1, Y_1) body zo snímku v referenčnej rovine a (x_0, y_0, z_0), (x_1, y_1, z_1) sú body mapy v priestore scény. Hĺbku bodu h_i za referenčnou rovinou označím ako výšku kamery mínus výšku z_i daného bodu nad podlahou . Ďalej je možné vyjadriť vzdialenosť na podlahe k danému bodu zo vzdialenosti v referenčnej rovine, tj. $vzd_i = h_i \cdot \|(X_i, Y_i)\|$. V tomto okamžiku vieme, že pozícia robota na podlahe je vzdialená vzd_0 od bodu $(x_0, y_0, 0)$ a vzd_1 od bodu $(x_1, y_1, 0)$. Prienikom týchto dvoch kružníc však môžu byť dve pozície robota $(r_1, s_1, 0)$ a $(r_2, s_2, 0)$. Ak body $(x_0, y_0, 0)$ a $(x_1, y_1, 0)$ ležia na jednej priamke prechádzajúcej takouto pozíciou, tak je pozícia iba jedna. Ak na takejto priamke neležia, tak sa vytvorí priamka z jednej pozície $(r_i, s_i, 0)$ do bodu $(x_0, y_0, 0)$ a skúma sa, či bod $(x_1, y_1, 0)$ leží napravo alebo naľavo od nej. Pokiaľ sa situácia zhoduje s rozložením bodov (X_0, Y_0) a (X_1, Y_1) v referenčnej rovine, tak je bod $(r_1, s_1, 0)$ jediným riešením. Situácia v druhom bode totiž musí byť opačná - strany sú zamenené zrkadlením, kde zrkadlom je rovina kolmá na podlahu a prechádzajúca bodmi $(x_0, y_0, 0)$ a $(x_1, y_1, 0)$. Dva dobre spárované body cez referenčnú rovinu tak dávajú jednoznačnú polohu kamery.

2.5 Priestor scény

Priestor scény sa chápe ako priestor, v ktorom sú definované všetky súradnice mapy. Jeho orientácia je určená podlahou, v ktorej ležia osi x a y. Celý problém vyhľadania pozície je možné zjednodušiť a definovať ako hľadanie zobrazenia medzi priestorom scény a priestorom robota. Takéto zobrazenie korešponduje so súradnicami robota aj jeho natočením na podlahe. Súradnice robota sú tak určené vektorom posunutia. A jeho orientácia na podlahe odpovedá matice otočenia Q_1.

$$A_{scn} = Q_1 \times A_{rbt} + OhniskoKamery_{scn} \tag{2.6}$$

$$A_{rbt} = Q_1^{-1} \times [A_{scn} - OhniskoKamery_{scn}] \tag{2.7}$$

$$Q_1 = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Pre úplnosť dodávam komplexné vyjadrenie rotačnej matice ľubovoľne orientovanej perspektívnej projekcie v našom modeli robota. Okrem natočenia kamery je zobrazenie definované vektorom posunutia, z ktorého je možné ľahko vyjadriť pozíciu ohniska kamery v priestore scény.

$$A_{scn} = Q \times [\lambda A_{img} - \vec{t}]; \quad \lambda A_{img} = Q^{-1} \times A_{scn} + \vec{t} \tag{2.8}$$

$$Q = Q_1 \times C_z \times Q_2 \times Q_3 \times C_y$$

$$Q = \begin{pmatrix} \cos\alpha\cos\beta - \sin\alpha\sin\beta\cos\delta & \cos\alpha\sin\beta + \sin\alpha\cos\beta\cos\delta & \sin\alpha\sin\delta \\ \sin\alpha\cos\beta + \cos\alpha\sin\beta\cos\delta & \sin\alpha\sin\beta - \cos\alpha\cos\beta\cos\delta & -\cos\alpha\sin\delta \\ -\sin\beta\sin\delta & \cos\beta\sin\delta & \cos\delta \end{pmatrix} \tag{2.9}$$

$$Q^{-1} = Q^T$$

$$\vec{t} = -Q^{-1} \times OhniskoKamery_{scn}$$

2.6 Zhrnutie

Perspektívna projekcia bodov scény do snímku kamery je určená maticou otočenia, vektorom posunutia a nakoniec vyhlásením daných súradníc za homogénne. Snímok má však už upravené súradnice bodov kalibráciou tak, aby odpovedal reprezentatívnym bodom (na rovine $z = 1$) v daných homogénnych súradniciach. V predchádzajúcich sekciách je dopodrobna rozobraná štruktúra matice otočenia, z ktorej je ľahko možné spätne vyjadriť informácie o jednotlivých fixných, či premenných parametroch ďalších výpočtov. Vektor $\vec{t}$ posunutia je často vyjadrený ako otočená pozícia kamery. Tu záleží len na poradí, v ktorom spojíme posun s otočením. Ak zvolíme pri projektovaní bodov najprv posun a až potom natočenie, tak treba presunúť ohnisko kamery do počiatku súradníc ($\lambda A_{img} = Q^{-1} \times (A_{scn} - OhniskoKamery_{scn})$). V opačnom prípade ($\lambda A_{img} = Q^{-1} \times A_{scn} + \vec{t}$) sa vyjadrí pozícia ohniska kamery ako $OhniskoKamery_{scn} = -Q \times \vec{t}$.

Nech je teda matica rotácie ortonormálna matica v nasledujúcom tvare:

$$Q = \begin{pmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{pmatrix}$$

Potom z nej možno vyčítať nasledujúce užitočné informácie o jednotlivých parametroch polohy kamery.

Projekcia bodu scény do snímku kamery:

$$\lambda A_{img} = Q^{-1} \times (A_{scn} - OhniskoKamery_{scn})$$

$$A_{img} = (\frac{q_{11}x + q_{21}y + q_{31}z + t_x}{q_{13}x + q_{23}y + q_{33}z + t_z}, \frac{q_{12}x + q_{22}y + q_{32}z + t_y}{q_{13}x + q_{23}y + q_{33}z + t_z}, 1) \qquad (2.10)$$

Priamka v scéne odpovedajúca pohľadu cez bod na obrázku:

$$p_{scn} = \lambda Q \times A_{img} + OhniskoKamery_{scn} \qquad (2.11)$$

Nasledujúce vyjadrenia uhlov β a δ platia iba, ak kamera nie je nasmerovaná kolmo nadol (resp. nahor). Inak matica vyjadruje iba jedno otočenie okolo vertikálnej osi. Takže jediný zmysel by mal iba uhol α, ktorý by sa v tomto krajnom prípade rovnal uhlu $\alpha = \arctan \frac{q_{21}}{q_{11}}$. Poprípade korekčný uhol β môže určovať odchýlku medzi natočením robota a natočením kamery. Avšak v tomto prípade ho nie je možné jednoznačne vyjadriť z matice natočenia. Je nutné, aby robot vedel jeho hodnotu pevne z kalibrácie. Pokiaľ však kamera pozerá šikmo na podlahu, tak je možné vyjadriť z matice Q jednotlivé interné parametre. A ďalší výpočet potom môže prebiehať efektívnejšie cez referenčnú rovinu.

Uhol korekcie otočenia snímku okolo osi pohľadu:

$$\beta = -\arctan \frac{q_{31}}{q_{32}} \qquad (2.12)$$

Vychýlenie smeru pohľadu od vertikálnej osi:

$$\delta = -\arctan \frac{q_{32}}{q_{33} \cdot \cos \beta} \qquad (2.13)$$

Orientácia robota:

$$\alpha = -\arctan \frac{q_{13}}{q_{23}}, (\text{resp.}\alpha = -\arctan \frac{q_{13}}{q_{23}} + \pi) \qquad (2.14)$$

Dôkaz týchto tvrdení plynie priamo z vyjadrenia matice rotácie vzorcom 2.9.

Dve kamery

Pri použití dvoch kamier je vhodné najprv správne namapovať body v oboch obrázkoch. Už pri známom nastavení kamier, tj. ich relatívnej polohe a natočení k druhej kamere predstavuje bod na jednom snímku priamku prechádzajúcu epipólou v druhom snímku. Epipóla je projekcia ohniska jednej kamery v snímku druhej. Pri mapovaní jedného bodu teda hľadám bod na takejto priamke, ktorý by mu mohol odpovedať. Tým dostávam priamo jeho relatívnu polohu bez toho, aby som musel vedieť napríklad jeho výšku v priestore scény.

Kapitola 3

Výpočet polohy robota na základe spárovaných bodov

Rozklad matice zobrazenia mapy do snímku kamery v kapitole 2 nám umožňuje využiť známe interné parametre kamery aj robota. Výpočet sa zúži priamo na neznáme premenné určujúce pozíciu robota. Hľadaná pozícia je tak určená namapovaním relatívneho priestoru robota do priestoru scény podľa vzťahu 2.7. Tento výpočtet pozície za pomoci známych interných parametrov je popísaný v sekcii 3.1.

Pri kalibrácii interných parametrov je však potrebné zobrazenie zrátať bez ich pomoci a na základe viacerých meraní tieto parametre dopočítať. Pre tieto účely možno použiť sofistikovanejšie algoritmy popísané v sekciách 3.3 a 3.4.

3.1 Metóda najmenších štvorcov nad referenčnou rovinou

Kalibrované body snímku sa na základe známych parametrov premapujú do referenčnej roviny vzťahom 2.5. Vstupom sú teda body v tejto rovine rovnobežnej s podlahou, spárované s bodmi scény. Po tomto kroku už naďalej netreba uvažovať žiadne interné parametre okrem výšky kamery nad podlahou. Body sa naďalej správajú ako perspektívne projektované, avšak ako keby pri pohľade kolmo nadol. Matica rotácie sa tým pádom zjednodušila

na

$$R = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$$

a posun je taktiež možný iba v rovine xy. Pri posune však treba byť opatrnejší, pretože body bližšie k podlahe sa pohybujú na referenčnej rovine pomalšie, ako body bližšie ku kamere. Koeficient tohoto pohybu je však veľmi ľahko vyjadriteľný z výšky kamery a 3D polohy posúvaných bodov po rovine rovnobežnej s podlahou.

Definuje sa teda vzťah pre spárovanú dvojicu bodov medzi referenčnou rovinou a scénou.

$$\begin{pmatrix} X_i \\ Y_i \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & t_x \\ \sin\alpha & \cos\alpha & t_y \end{pmatrix} \times \begin{pmatrix} x_i \\ y_i \\ \lambda_i \end{pmatrix} \tag{3.1}$$

Pri tomto vyjadrení treba nahliadnuť, že $(X_i, Y_i)^T$ je bod v referenčnej rovine (viď sekciu 2.4) a $(x_i, y_i, 1)^T$ je jemu prislúchajúci bod v scéne projektovaný do referenčnej roviny ukotvenej v počiatku súradníc scény O. A jeho koeficient mierky λ_i vyjadruje 1/(rozdiel medzi výškou kamery a jeho pôvodnou výškou bodu v scéne). Pohyb robota, si je vhodné predstaviť ako pohyb týchto projektovaných bodov scény. Ohnisko projekcie referenčnej roviny tak zostáva totožné pre oba smery prevodu bodov.

Vzdialenosť v referenčnej rovine bodov scény rovnakej výšky je treba rátať ako ich vzdialenosť v scéne násobenú koeficientom mierky λ_i.

Metóda najmenších štvorcov je založená na minimalizácii chyby výslednej transformácie. Funkcia chyby sa definuje ako súčet vzdialenosti bodov v referenčnej rovine. Vzdialenosti sa teda počítajú medzi bodmi $(X_i, Y_i)^T$, ktoré sú do tejto roviny priamo projektované z obrázku a z im spárovanými bodmi $(x_i, y_i, 1)^T$, ktoré sú transformované zo scény.

$$\phi(\alpha, t_x, t_y) = \sum_{i=1}^{n} \left\| \begin{pmatrix} X_i \\ Y_i \end{pmatrix} - \begin{pmatrix} \cos\alpha & -\sin\alpha & t_x \\ \sin\alpha & \cos\alpha & t_y \end{pmatrix} \times \begin{pmatrix} x_i \\ y_i \\ \lambda_i \end{pmatrix} \right\|^2$$

$$\phi(\alpha, t_x, t_y) = \sum_{i=1}^{n} \left[\begin{pmatrix} (X_i - x_i\cos\alpha + y_i\sin\alpha - \lambda_i t_x)^2 + \\ +(Y_i - x_i\sin\alpha - y_i\cos\alpha - \lambda_i t_y)^2 \end{pmatrix} \right] \tag{3.2}$$

jej parciálne derivácie sú:

$$\begin{aligned}
\frac{\partial\phi(\alpha,t_x,t_y)}{\partial\alpha} &= 2\sum_{i=1}^{n}\left[\begin{array}{l}(x_i\sin\alpha+y_i\cos\alpha)(X_i-x_i\cos\alpha+y_i\sin\alpha-\lambda_i t_x)+\\+(-x_i\cos\alpha+y_i\sin\alpha)(Y_i-x_i\sin\alpha-y_i\cos\alpha-\lambda_i t_y)\end{array}\right]\\
\frac{\partial\phi(\alpha,t_x,t_y)}{\partial t_x} &= 2\sum_{i=1}^{n}(-\lambda_i)(X_i-x_i\cos\alpha+y_i\sin\alpha-\lambda_i t_x)\\
\frac{\partial\phi(\alpha,t_x,t_y)}{\partial t_y} &= 2\sum_{i=1}^{n}(-\lambda_i)(Y_i-x_i\sin\alpha-y_i\cos\alpha-\lambda_i t_y)
\end{aligned} \tag{3.3}$$

Funkcia je hladká na celom svojom definičnom obore. V jej lokálnych extrémoch sú jej parciálne derivácie rovné nule. Ukazuje sa, že situácia, keď sú tieto derivácie nulové sú práve dve a z povahy funkcie bude práve jedna z nich minimom. Riešenie sústavy nulových parciálnych derivácií je riešením nasledujúcej sústavy rovníc.

$$\begin{aligned}
t_x &= \tfrac{K_1-K_2\cos\alpha+K_3\sin\alpha}{K_5}\\
t_y &= \tfrac{K_4-K_2\sin\alpha-K_3\cos\alpha}{K_5}\\
K_6\sin\alpha+K_7\cos\alpha &= 0
\end{aligned} \tag{3.4}$$

Výsledkom sú dve protikladne orientované riešenia. Treba poznamenať, že funkciu inverzného tangensu možno zapísať v tvare pre oddelené zložky protiľahlej a priľahlej strany. Pre takýto zápis je jej možné ľahko dodefinovať krajné prípady hodnotou $\pi/2$ alebo $-\pi/2$. A to, či je robot orientovaný vpred alebo vzad sa musí určiť dosadením týchto dvoch možností do funkcie chyby a vziať takú, kde je chyba menšia. Dvojznačnosť riešenia vzniká položením derivácie uhlu rovnej nule. Situácia nastáva v dvoch prípadoch. A to, keď je natočenie optimálne a keď je natočenie opačné, tj. najhoršie.

Pre úplnosť - koeficienty výpočtu sústavy rovníc 3.4 :

$$\begin{aligned}
K_1 &= \sum_{i=1}^{n}\lambda_i X_i\\
K_2 &= \sum_{i=1}^{n}\lambda_i x_i\\
K_3 &= \sum_{i=1}^{n}\lambda_i y_i\\
K_4 &= \sum_{i=1}^{n}\lambda_i Y_i\\
K_5 &= \sum_{i=1}^{n}\lambda_i^2\\
K_6 &= \sum_{i=1}^{n}(X_i x_i+Y_i y_i)-\tfrac{1}{K_5}\sum_{i=1}^{n}\lambda_i(K_1 x_i+K_4 y_i)\\
K_7 &= \sum_{i=1}^{n}(X_i y_i-Y_i x_i)-\tfrac{1}{K_5}\sum_{i=1}^{n}\lambda_i(K_1 y_i-K_4 x_i)
\end{aligned}$$

Treba však poznamenať, že na takýto výpočet pozície sú nutné minimálne dva rôzne páry bodov, kroré je možné zobraziť v referenčnej rovine. Ak je takýchto párov viac, tak by mali zachovávať topológiu v priestore tak, aby nebolo K_6 a K_7 naraz rovné nule.

Reprezentáciu výsledku si možno predstaviť ako otočenie a potom posunutie už v otočenom priestore. Ak sa požaduje najprv posunutie a až

potom otočenie, tak stačí vektor posunutia otočiť späť o daný uhol $-\alpha$. Inak povedané $Rx + t = R \times (x + R^{-1}t)$, kde R je matica otočenia o uhol α.

3.2 Váhy bodov v referenčnej rovine

Keďže transformácia obrázku z kamery do referenčnej roviny obnáša určitú chybu, tak je dobré jednotlivým bodom $A_i = (X_i, Y_i)$ priradiť váhu $w_i(A_i)$. Tento koeficient vyjadruje to, ako sú body podstatné pri výpočte polohy metódou najmenších štvorcov. Úprava metódy spočíva v pridaní tohoto koeficientu ako násobiteľa členov v každej sume vyjadrenej v predchádzajúcej sekcii.

Vyjadrenie chyby spočíva v úvahe, že čím je bod bližšie k horizontu, tým sa dopúšťa väčšej nepresnosti pri transformácii do referenčnej roviny. Nech sa uvažuje iba chyba, ktorá vzniká medzi bodmi na jednom riadku obrázku. Odpočítaním projekcie $A_{cam} = (u, v)$ od $A'_{cam} = (u + \Delta, v)$ do referenčnej roviny (podľa zvťahu 2.5: $A_{ref} - A'_{ref} = (\Delta/(v \sin\delta + \cos\delta), 0, 1)$) je vidieť, že táto chyba je na celom riadku rovnaká. Ak sa nazrie na situáciu ako na pohľad na dve rovnobežky, ktoré sa spájajú na horizonte, tak je zrejmé že ich vzdialenosť na obrázku z kamery sa zmenšuje lineárne až po riadok, ktorý predstavuje horizont. Riadok s najmenšou chybou by v tomto modeli predstavoval riadok, v ktorom by sa nachádzal kolmý pohľad na rovinu podlahy. Samozrejme, že ani pohľad na horizont ani pohľad na bod pod kamerou nemusí byť v obraze zahrnutý. Stačí tieto y-ové súradnice v priestore kamery zrátať analiticky. Ak neexistuje horizont v rovine snímku kamery, tak to znamená, že kamera je už nastavená ideálne, kolmo na podlahu a jej priestor snímku sa zhoduje s priestorom referenčnej roviny. V tomto prípade nie je potrebný žiaden prevod, takže chyba sa nezapočítava.

Horizontálne váhy tak môžu byť lineárne závislé iba od výšky bodu na snímku z kamery, tak aby dané krajné výšky predstavovali váhy 1 a 0.

Vertikálnu chybu je možné zrátať pre konkrétny prípad ako rozdiel medzi projektovaným bodom do referenčnej roviny a bodom, ktorý vznikol jeho vertikálnym posunutím v snímku a projektovaním do referenčnej roviny. Bohužiaľ tu už nevzniká tak pekné rozloženie chyby ako pri chybe horizontálnej. V smere od kamery je chyba na referenčnej rovine väčšia ako v smere ku kamere. Pre jednoduchosť je však možné chybu odhadnúť chybou horizontálnou.

Ďalšie váhy, ktoré je možné do metódy najmenších štvorcov zarátať sú váhy z párovania. Tieto váhy určujú dôveru k daným párom zahrnutým vo

výpočte. Sú taktiež volené z intervalu $(0, 1]$, kde 1 znamená úplnú dôveru k danému páru. Ak tieto váhy označím ako w_i a váhy nepresnosti v zobrazení označím ako h_i, tak rovnica chyby bude upravená na tvar:

$$\phi(\alpha, t_x, t_y) = \sum_{i=1}^{n} \left[w_i h_i \cdot \begin{pmatrix} (X_i - x_i \cos\alpha + y_i \sin\alpha - \lambda_i t_x)^2 + \\ +(Y_i - x_i \sin\alpha - y_i \cos\alpha - \lambda_i t_y)^2 \end{pmatrix} \right] \tag{3.5}$$

To sa prejaví nakoniec vo výpočte koeficientov, ktoré budú analogicky pozmenené k tejto rovnici chyby.

3.3 POSIT

Zatiaľ, čo vo výpočte cez referenčnú rovinu nám postačujú dva páry bodov (Veta 2) v referenčnej rovine, pri všeobecnom prípade perspektívnej projekcie je potrených párov aspoň 4. A to také aby neležali na jednej rovine.

Na takýto výpočet existuje viacero postupov. Asi najznámejšie sú POSIT, DeMenthon & Davis, 1992 [1] a Lineárny algoritmus [6]. Dané výpočty sa využívajú pri vkladaní a uchopovaní objektov zo snímkov priestorovým modelovaním.

POSIT je založený na iteračnom odhade rozdielov jednotlivých bodov zobrazených slabou a silnou projekciou. Slabá projekcia najprv projektuje body kolmo na nejakú rovinu rovnobežnú s rovinou snímku kamery. A až potom pomocou jednotnej zmeny mierky sa tieto body zobrazia v snímku kamery. V každom kroku sa tak vyjadrí matica slabej projekcie cez nejaký zvolený bod scény. Jej normalizovaný tvar konverguje k matici otočenia pri klasickej projekcii. Z nej sa určia nové rozdiely pre jednotlivé body.

Nech $M_i = (x_i, y_i, z_i)_{i=0}^{N}$ sú body v scéne a $d_i = (X_i, Y_i)_{i=0}^{N}$ sú po rade im odpovedajúce body na kalibrovanom snímku kamery (ohnisková vzdialenosť je 1). Ako pevný bod je zvolený napríklad M_0. Jemu odpovedajúci bod na obrázku je bod d_0. Pomocou hĺbky Z_0 (vzdialenosti M_0 od roviny snímku) tohoto projektovaného bodu za snímkom je možné definovať slabú projekciu cez nejakú rovinu prechádzajúcu bodom M_0. Na rozdiel od silnej perspektívnej projekcie (napr. $(X_i, Y_i) = (x_i/z_i, y_i/z_i)$) slabá projekcia používa spoločnú hĺbku (napr. $(X_i, Y_i) = (x_i/Z_0, y_i/Z_0)$). Umiestnenie bodov do nejakej roviny rovnobežnej s rovinou snímku prechádzajúcej M_0 je možné zapísať ako ich natočenie okolo M_0 ($R \times (M_i - M_0) + M_0$, kde R je matica daného natočenia) a nahradenie ich poslednej súradnice hĺbkou Z_0. Projektovaním slabou projekciou teda vznikajú body na snímku $((1/Z_0)R_1 \times (M_i - M_0) + X_0, (1/Z_0)R_2 \times (M_i - M_0) + Y_0)$, kde R_i je i-ty riadok

matice R. Rozdiel medzi týmito bodmi a bodmi vzniknutými silnou projekciou sa v ďalších krokoch vyjadrí ako $\epsilon_i \cdot d_i$, kde $\epsilon_i = (1/Z_0) \cdot R_3 \times (M_i - M_0)$. V jednom kroku sa tak vyjadria koeficienty prvých dvoch riadkov matice P (podľa rovníc 3.6), ktorá zastupuje maticu $(1/Z_0)R$. Potom sa spočítajú nové korekčné hodnoty ϵ_i a postup sa opakuje pokiaľ sa korekčné hodnoty dostatočne zmenili.

$$\begin{aligned} P_1 \times (M_i - M_0) &= (1 + \epsilon_i) \cdot X_i - X_0 \\ P_2 \times (M_i - M_0) &= (1 + \epsilon_i) \cdot Y_i - Y_0 \end{aligned} \tag{3.6}$$

$$\epsilon_i := (1/\|P_1\|) \cdot (P_1 \times P_2) \times (M_i - M_0)$$

Pretože platí $P = (1/Z_0)R$ a R je ortonormálna matica otočenia, tak sa nakoniec normalizáciou P vyjadrí matica R a Z_0. Poloha ohniska kamery je vektor $R^{-1} \times (X_0 * Z_0, Y_0 * Z_0, Z_0)^T - M_0$. Všetky ostatné parametre pri prechode na prepočet cez referenčnú rovinu je možné vyjadriť z matice R pomocou vzťahov v sekcii 2.6. Algoritmus POSIT je implementovaný v knižnici OpenCV [4].

3.4 Linear Algorithms

Ďalším algoritmom, ktorý dokáže zrátať pozíciu kamery aj pokiaľ nepoznáme jej výšku nad podlahou ani jej natočenie vzhľadom k podlahe je 4-bodový lineárny algoritmus na výpočet obecnej polohy kamery pri perspektívnej projekcii. Obecne je ho možné rozšíriť na 5 až N bodov. Algoritmus je možné napísať na 25 riadkov kódu v prostredí Mathematica, ako uvádzajú Quan & Lan, 1999 [6].

Nech to, čo sa pokúšam zrátať sú vzdialenosti x_i od ohniska kamery k jednotlivým bodom scény M_i. Vychádza sa z toho, že na obrázku je možné ľahko vyjadriť uhol pohľadu (γ_{ij}) medzi dvomi pozorovanými bodmi. A že je známa vzdialenosť medzi týmito bodmi v scéne (d_{ij}). Potom možno pre vzdialenosti ohniska kamery k týmto bodom (x_i a y_i) utvoriť vzťah pomocou kosínovej vety: $x_i^2 + x_j^2 - 2x_i x_j \cos\gamma_{ij} = d_{ij}^2$. Pri každej trojici bodov je možné eliminivať dve vzdialenosti ich vyjadrením pomocou tretej premennej. Tieto tri rovnice tak zapísať jednou ako: $g(x) = a_5 x^4 + a_4 x^3 + a_3 x^2 + a_2 x^1 + a_1 = 0$, kde $x := x_i^2$.

Už pre štyri body tak vzniká sústava

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \end{pmatrix} \times \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \\ x^4 \end{pmatrix} = 0$$

Túto sústavu je možné riešiť pomocou SVD (Singular Value Decomposition). Detaily výpočtu tejto sústavy sú uvedené napríklad v [6, Quan & Lan, 1999]. Z x je možné dopočítať vzdialenosť ohniska k i-temu bodu mapy $x_i = +\sqrt{x}$. A následne vzdialenosti k ostatným bodom.

Výsledkom však sú vzdialenosti k jednotlivých bodov od ohniska kamery. Tj. ich umiestnenie v relatívnom priestore kamery je $(X_i * Z_i, Y_i * Z_i, Z_i)$, kde (X_i, Y_i) sú súradnice daného bodu na snímku. Metódou najmenších štvorcov je tak možné vyjadriť maticu rotácie Q aj s polohou ohniska kamery priamo vyjadrením koeficientov rovnice 2.8, kde $\lambda_i = Z_i$. Všetky ostatné parametre pri prechode na prepočet cez referenčnú rovinu je možné vyjadriť z matice Q pomocou vzťahov v sekcii 2.6.

Kapitola 4

Spracovanie Obrazu

Snímok objektu alebo scény vyžaduje špeciálne spracovanie, ktoré je upravené tak, aby dokázalo v uvedených podmienkach z obrázku vytiahnuť čo najviac užitočných informácií. Často býva výsledok závislý nielen na farbách a kontraste objektov v scéne, ale aj na konkrétnom osvetlení. Pri metódach, ako je prahová segmentácia (sekcia 4.2), alebo klasifikácia farieb je veľmi pravdepodobné, že napríklad dôjde k stratám informácií pri zmene intenzity svetelných zdrojov. Naopak pri spracovaní podľa lokálnych zmien v obrázku, čo sú hlavne hranové detektory (sekcia 4.3), je závislosť na type a intenzite osvetlenia omnoho slabšia. Rýchle a do značnej miery robustné spracovanie snímku je možné docieliť vďaka využitiu histogramov. To je uvedené v sekcii 4.4. Oblasti uchopené na základe farieb je treba správne lokalizovať. Jednoduché uchopenie pomocou ich ťažiska zo sebou nesie určité nepresnosti. Ich popis a korekcia je popísaná v sekcii 4.5. Problémami, s ktorými sa na najnižšej úrovni spracovania obrázku bojovať nedá, sú tiene a prekrývanie objektov v scéne. Keďže pri spracovaní obrázku ešte nie sú známe pozície ani tvar objektov, tak je možné, len aby sa dané segmenty simulovali v ďalších fázach spracovania podľa predpokladaného pohľadu do scény. Tieto problémy je teda možné zaradiť pod zobrazovanie scény a spracovanie dotazov do mapy. Vzhľadom na rozsiahlosť tejto práce som sa bohužiaľ nezaoberal tieňmi, odrazmi ani priehľadnými alebo lesklými telesami.

4.1 Detekcia hrán

Hrana na obrázku sa definuje ako náhly prechod intenzity v danom smere. Pre detekciu hrán sú vstupom obrázky v odtieňoch jednej farby. Preto je vhodné zvážiť výber transformácie plne farebného obrázku do obrázku v stupňoch šedi popísanú v sekcii 4.4.

Cannyho hranový detektor

Najznámejší a najpoužívanejší hranový detektor. Jeho pomenovanie podľa jeho objaviteľa Johna Cannyho z roku 1986, ktorý definoval požiadavky pre hranový detektor následovne.
1. Hranový detektor, by mal byť odolný voči šumu na obrázku
2. Mal by určovať správne pozíciu hrany
3. A nemal by mať násobné odozvy na jednu reálnu hranu

Šumu na obrázku sa zbavuje pomocou konvolúcie s gausiánom. A samotné hrany potom detekuje pomocou filtru, ktorý vzniká deriváciou tohoto gausiánu. Po týchto dvoch konvolúciách vzniká ohodnotenie bodov, ktorého absolútna hodnota odpovedá ich vôle byť hranou. Na ďalšie spracovanie sa tak zvolí horný prah pre dané absolútne hodnoty ohodnotenia, ktorý definuje isté hrany. A dolný prah, ktorý definuje neisté hrany. Tie sa stávajú hranami až v prípade, keď hraničia s už určenými hranovými bodmi.

Keďže 2D konvolúcia je náročná na výpočtový čas procesoru, tak sa spravidla používa rozdelenie konvolučných masiek na x-ovú a y-ovú zložku. Hodnoty hrán sa tak zrátajú oddelene vo vertikálnom M_x aj v horizontálnom smere M_y. Ich spojenie sa definuje ako $M(x,y) = \sqrt{M_x^2(x,y) + M_y^2(x,y)}$. Klasifikáciu neistých hrán je možné robiť rekurziou.

Cannyho hranový detektor je veľmi známy a je imlementovaný napríklad v knižnici OpenCV [4]. Dané požiadavky pre hranový detektor je však možné riešiť aj inými spôsobmi. Napríklad Shen-Castan detektor je v odolnejší na šum a lepšie lokalizuje hrany avšak je náchylnejší v násobnej odozve jednej hrany, ako je uvedené v J.R.Parkerovi,1997 [5].

4.2 Prahová Segmentácia

Segment je súvislá množina pixelov obrázku, ktoré vypĺňajú nejakú špecifickú oblasť alebo majú podobné farby. Rozdelenie bodov do segmentov je prvý veľký krok v spracovaní informácií z obrázku. Nie je ho však nutné vždy

Algoritmus 1: Cannyho hranový detektor

Data:
I - pôvodný obrázok
σ - stredná hodnota rozptylu pre určenie gausiánu
T_{high} - dolný prah pre určenie istých hrán
T_{low} - dolný prah pre určenie neistých hrán **Result**:
I - hrany obrázoku I

```
begin
    vyjadrenie 1D konvolučných masiek gausiánu: G_x, G_y;
    vyjadrenie 1D konvolučných masiek derivácie gausiánu: G'_x, G'_y;
    I_x = konvolúcia I s G_x po riadkoch;
    I_y = konvolúcia I s G_y po stĺpcoch;
    M_x = konvolúcia I s G'_x po riadkoch;
    M_y = konvolúcia I s G'_y po stĺpcoch;
    M(x,y) = sqrt(M_x^2(x,y) + M_y^2(x,y));
    for (x,y) : M(x,y) ≥ T_high do
        E(x,y) = hrana;
    for (x,y) : M(x,y) ≥ T_low do
        if E(x,y) ≠ hrana then
            urči ako hranu, pokiaľ je možné určiť susedný pixel ako
            hranu (rekurzívna funkcia na klasifikáciu neistých hrán);
    return E
end
```

aplikovať, pretože existuje mnoho ďalších metód ako porovnávať resp. klasifikovať časti obrázkov.

Prahovú segmentáciu čierno-bieleho obrázku si možno predstaviť ako radikálne zmenšenie počtu farieb obrázku tak, aby sa stratilo čo najmenej informácií. Dané zmenšenie počtu odtieňov môže mať globálny aj lokálny charakter. Pri globálnom thresholding-u sa určia disjuktné intervaly farieb pre konkrétne segmenty na celom obrázku, zatiaľ čo pri lokálnom sa prahy určujú adaptívne podľa okna, ktoré predstavuje menší výrez. Samotné prahovanie spočíva v zaradení pixelov obrázku do daných intervalov, čím sa definujú jednotlivé segmenty ako súvislé oblasti z jedného intervalu.

Pri akejkoľvek presnej hodnote prahov (hraníc intervalov farieb), vzniká tzv. kvantizačný šum. Daný nežiadúci efekt možno pozorovať na digitálnych fotografiách alebo digitálnom videu pri slabom počte farieb. Na väčších jednoliatych plochách sa začnú objavovať nežiadúce hrany, pretože

farby nepostačujú na plynulý prechod odtieňov. Dopad býva až taký veľký, že sa segmentácia prahovaním stáva nepoužiteľnou.

Intervaly farieb pri farebnom obrázku sú kvádre v nejakom farebnom priestore. Základným takýmto priestorom je farebný model RGB, kde jednotlivé osi určujú intenzitu elementárnej farby. Interval farby v tomto priestore je takmer nepoužiteľný. Je buď moc malý, čo nemusí postačovať rôznym intenzitám ani uhľom pohľadu na danú jednofarebnú plochu. Alebo nesie informácie o viacerých farbách, ktorými rozhodne skúmaná plocha nedisponuje. Ďalšou možnosťou je napríklad použiť model HSV, kde je jednou osou farebnosť. No podobný problém nastáva pri klasifikovaní bielej alebo čiernej farby, ktoré na tejto ose nemajú svoje význačné miesto.

4.3 Hranová Segmentácia

U globálneho prahovania aj segmentácie klasifikáciou farieb hrá príliš veľkú rolu model osvetlenia scény. Tomu sa dá vyhnúť napríklad pri definovaní segmentov ako súvislých oblastí medzi hranami. Hrany však zďaleka nemusia byť vždy úplné a malé vynechanie môže spôsobiť zliatie segmentov dohromady. Pokiaľ však môžeme z nejakého dôvodu dôverovať hranovému detektoru, tak sa dajú očakávať výborné výsledky. A to hlavne pri dostatočne kontrastných hranách segmentov.

4.4 Histogram

Histogram určuje početnosť farieb na určitej oblasti snímku. Možno si ho predstaviť ako funkciu, ktorej definičným oborom sú farby a hodnota je priamo úmerná počtu bodov danej farby na danej oblasti snímku. Pri spracovaní snímku je možné použiť histogram na segmentáciu, na klasifikáciu segmentov z hranovej segmentácie, alebo na trakovenie pozície segmentov pomocou Mean-Shift algoritmu.

Zrovnávacie histogramy sú uložené v mape ako vlastnosti jednotlivých segmentov. Je vhodné ich vytvárať zo snímkov obsahujúcich iba dané segmenty pod rôznymi pohľadmi. Pokiaľ nie je daná možnosť, tak je ich možné dogenerovať v predspracovaní mapy z textúr segmentov.

Pre obrázky v odtieňoch šedi je histogramom jednorozmerná funkcia. Avšak pri plne farebnom obrázku je histogram funkciou troch zložiek farby. Vo väčšine prípadov nie je nevyhnutné udržiavať takéto veľké trojrozmerné histogramy. Je treba si rozmyslieť ako sa daná oblasť líši od svojho okolia.

Pokiaľ nie je dôležitá sýtosť ani intenzita farby, tak je možné namiesto trojrozmerného modelu histogramu vziať iba prvú zložku farby vo farebnom priestore HSV. Priestor farieb HSV bol vyvinutý pre kompresie farebných obrázkov a videa. Preto je jeho snahou extrahovať informácie o farbách do jednej osi nejakého 3D priestoru farieb. Jeho osi tvoria farebnosť, sýtosť a hodnota. Hodnota farebnosti je na identifikáciu sýtej farby najdôležitejšia. Pokiaľ je však klasifikovaná farba odtieňom šedi, tak je pre ňu táto zložka bezvýznamná. V tomto prípade je vhodnejšie použiť prevod do čierno-bieleho obrázku v stupňoch šedi.

Segmentáciu na základe normalizovaných histogramov segmentov prebieha ako klasifikácia každého bodu obrázku podľa farby. Bod je nakoniec priradený segmentu, kde má v histograme najväčšiu hodnotu.

Klasifikácia segmentov detekovaných podľa ich tvaru (napríklad hranovou segmentáciou) je realizovateľná použitím vhodnej metódy na porovnanie histogramov. Pri porovnaní je dôležité brať ohľad hlavne na rozloženie lokálnych extrémov v histograme. Preto je lepšie používať metódy ako je korelácia alebo Battacharyyaova vzdialenosť. Koreláciu dvoch histogramov je možné vyjadriť ako

$$d(H_1, H_2) = \frac{\sum_{i=1}^{N} H_1'(i) \cdot H_2'(i)}{\sqrt{\sum_{i=1}^{N}[H_1'(i)^2] \cdot \sum_{i=1}^{N}[H_2'(i)^2]}}$$

kde

$$H_k'(i) = H_k(i) - \frac{1}{N}\sum_{j=1}^{N} H_k(j)$$

Pokiaľ sa nazerá na celý histogram ako na jednotkový vektor v N-rozmernom priestore, tak druhou mocninou Battacharyyaovej vzdialenosti medzi takýmito vektormi je jedna mínus odmocnina kosínusu uhlu medzi týmito vektormi.

$$d(H_1, H_2) = \sqrt{1 - \sum_{i=1}^{N} \sqrt{H_1(i) \cdot H_2(i)}}$$

Treba poznamenať, že takto ide danú vzdialenosť počítať iba pre normalizované histogramy.

Na klasifikáciu je vhodné nad mapou vytvoriť množiny segmentov s podobnými histogramami. Jednu takúto množinu potom stačí identifikovať jedným histogramom, ktorý predstavuje jej strednú hodnotu.

Histogram je taktiež výhodné používať pri trakovaní pozície segmentov, na základe ktorých je možné trakovať samotnú pozíciu robota. Využíva sa pri tom spätná projekcia histogramu do obrázku kamery. Výsledkom je

obrázok, ktorého body obsahujú namiesto farby hodnotu, ktorá odpovedá danej farbe v histograme. Vznikne tak rozloženie s hustotou hodnôt koncentrovanou do segmentov skúmanej farby. Ak je vhodne zvolené lokálne okno na obrázku, tak pomocou váženého priemeru súradníc jeho bodov poloha stredu okna konverguje do ťažiska segmentu. Takýto iteračný postup je známy ako Mean-Shift algitmus a je bližšie popísaný v sekcii 7.1.

4.5 Ťažiská segmentov

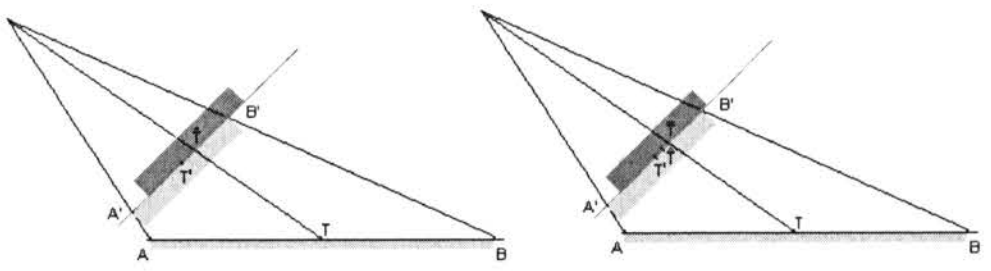
Po detekovaní segmentov na obrázku je ich poloha väčšinou popísaná iba jedným bodom. Daným bod je napríklad ťažisko segmentu. Je to stred segmentu na obrázku. Jeho výhodou je rýchly a robustný výpočet ako aritmetický priemer súradníc bodov ležiacich v danom segmente na snímku z kamery. V prípade, že chceme takéto ťažisko na snímku vyjadriť projekciou segmentu z mapy pri známej pozícii je situácia trocha komplikovanejšia, pretože perspektína projekcia nezobrazuje ťažiská v mape do ťažísk obrazu. Dané ideálne zobrazenie ťažísk by nastávalo iba v situácii ak je rovina snímku rovnobežná s rovinou segmentu v scéne. Riešenie všeobecného prípadu naskytujú priamo triangulizované mapy, ktorých elementárnymi segmentami sú trojuholníky. Výpočet ťažiska takéhoto segmentu na snímku spočíva v atirmetickom priemere súradníc jeho troch vrcholov projektovaných do snímku. Ak by sme chceli segment obrázku identifikovať s viacerými trojuholníkovými segmentami mapy, tak stačí tieto neprekrývajúce sa segmenty na snímku spojiť váženým priemerom. Váhou ťažiska elementárneho segmentu je jeho obsah na obrázku. A výsledným ťažiskom tohoto zhluku segmentov je takto vážený aritmetický priemer ťažísk spojených elementárnych trojuholníkových segmentov na obrázku.

Spätnú identifikáciu takéhoto ťažiska s nejakým bodom mapy je možné previesť ako vektor rozdielu medzi projektovaným ťažiskom z mapy a daným ťažiskom na snímku. Alternatívou je počítať tento rozdiel v mape. A to pri rovinných segmentoch ako prienik lúča vychádzajúceho z ohniska kamery cez bod vyjadrujúci ťažisko na snímku snímku, s rovinou segmentu. Vektor posunu ťažiska v mape by bol rozdiel takéhoto bodu s bodom, ktorý vyjadruje ťažisko v mape. Tieto dva postupy sú si ekvivalentné. S tým rozdielom, že ten prvý je ľahšie implementovateľný aj pre nerovinné zložené segmenty mapy.

Pri neznámej polohe kamery je nutné postupovať iteračne. Začína sa stotožnením ťažísk obrazu s ťažiskami mapy. Po výpočte pozície na základe

takýchto párovaní sa upresnia pozície ťažísk pomocou spätnej identifikácie pri tejto projekcii. Je dobré si uvedomiť, že chyba je tým väčšia, čím pozerám na rovinu segmentu pod väčším uhlom. Zároveň je závislá od veľkosti segmentu na obraze. Keďže presné ťažisko z mapy projektované správnou projekciou do snímku neopustí konvexný obal segmentov tak je daná približná projekcia natoľko blízka správnej projekcii, že konvexné obálky daných segmentov medzi nimi sa prekrývajú v snímku aspoň z jednej štvrtiny. Ďalšou iteráciou cez upresnené pozície ťažísk algoritmus postupuje podobne ako POSIT popísaný v sekcii 3.3. S tým rozdielom, že neupresňuje chybu medzi slabou a silnou projekciou jednotlivých bodov v mape, ale chybu polohy ťažísk na snímku.

Obrázok 4.1: Korekcia ťažísk segmentov



Nech T_i označím ako ťažiská v mape $\hat{T}_i$ ako ťažiská z obrazu odpovedajúcich segmentov a m_i ich korekčné posuny zo spätnej identifikácie pri poslednej určenej približnej projekcii. Potom je možné zrátať z párovania $< T_i, \hat{T}_i + m_i >$ novú presnejšiu približnú projeckiu charakterizovanú maticou otočenia R a posunom $\vec{t}$. Projekcia je určená tak, aby sa blížila k rovniciam projekcie pre jednotlivé ťažiská $\hat{T}_i + m_i = (1/(R_3 T_i + t_z))(R_1 T_i + t_x, R_2 T_i + t_y)$. Ako už bolo poznamenané projekcia ťažiská z mapy máličko posúva. Ťažiská na obrázku sa teda vyjadria v danej projekcii presne projektovaním segmentov. V prípade trojuholníka je možné takéto ťažiská vyjadriť ako $T'_i = (A'_i + B'_i + C'_i)/3$, kde A'_i, B'_i, C'_i sú projektované vrcholy A_i, B_i, C_i segmentu číslo i. Ďalej zvolíme $\epsilon_{A_i} = \frac{R_3 A_i + t_z}{R_3 T_i + t_z}$, $\epsilon_{B_i} = \frac{R_3 B_i + t_z}{R_3 T_i + t_z}$ a $\epsilon_{C_i} = \frac{R_3 C_i + t_z}{R_3 T_i + t_z}$. Zo zvolenia projekcie vyjadríme $\hat{T}_i + m_i = (\epsilon_{A_i} A'_i + \epsilon_{B_i} B'_i + \epsilon_{C_i} C'_i)/3$. No a nová korekcia ťažiska je určená rozdielom týchto vyjadrení ťažísk v projektovanej rovine ako $T'_i - \hat{T}_i$. A zmenu korekcie ťažiska v tomto kroku je možné vyjadriť ako $\Delta m_i = ((1 - \epsilon_{A_i}) A'_i + (1 - \epsilon_{B_i}) B'_i + (1 - \epsilon_{C_i}) C'_i)/3$. A za novú odhadnutú pozíciu ťažisiek segmentov mapy na obraze z kamery sa vyhlásia body $\hat{T}_i + m_i + \Delta m_i$. Ak je táto zmena Δm_i dosť malá aby vyhovovala požadovanej presnosti určenia projekcie, tak iterácie končia. Keďže ťažiská sa v danom pohľade posúvajú vždy do korektných pozícií, tak zvolenie ďalšieho pohľadu vychádza z lepšieho zvolenia párovaných bodov.

Preto je nová projekcia lepšia ako projekcia pred korekciou pozícií ťažísk. V praxi sa ukazuje, že na určenie pozície segmentov rovnomerne rozložených na obrázku pri veľkosti segmentov nepresahujúcich 1/5 šírky obrazu zväčša postačujú jedna až dve iterácie korekcie ťažiska.

Na obrázku 4.1 je zobrazený počiatočný stav a medzistav korekcie pri zjednodušenom projektovaní 2D bodov na priamku. Popis bodov odpovedá označeniu použitom pri výpočte, až na bod $\overline{T}$, ktorý tu označuje pozíciu $\hat{T} + m$. T je ťažisko v mape, $\hat{T}$ je pôvodné ťažisko z obrázku kamery a T' je korektné ťažisko pri danom pohľade. Zmenu korekcie ťažísk si možno všimnúť ako vzdialenosť $T' - \hat{T}$.

Kapitola 5

Párovanie identických bodov

V prípade, že robot nemá jednu približnú pozíciu, tak je potreba problém párovania riešiť komplexnejšie. Výpočty pozície spoliehajú najmä na klasifikáciu segmentov. Totiž ak by sa stalo, že kamera vidí aspoň dva segmenty rôznych typov, ktoré majú v mape iba jeden výskyt, tak je nie je nutné párovať a pozíciu je možné získať priamo na základe daného prostého priradenia segmentov do mapy. Párovanie je nutné u typov segmentov rozpoznaných v obraze a majúcich v mape viacero výskytov. Ak prebieha klasifikácia podľa farby, tak si možno predstaviť ako typ napríklad segmenty červenej farby. Pri jedinom takomto segmente v mape je priradenie jednoznačné. Ak je ich viac, tak je možné červené segmenty klasifikovať podľa iných príznakov, alebo skúšať rôzne priradenia množiny červených segmentov mapy k danému segmentu v obraze. V takomto prípade hovoríme o množine identických bodov. A ich kombinovanie medzi obrázkom a mapou popisuje celá táto kapitola.

Inou situáciou je znalosť približnej polohy robota v mape, ktorú chceme upresniť. T tomto prípade je možné využiť rýchlejšie metódy registrácie bodov mapy k bodom obrázku. Body na párovanie s mapou je vhodné brať aj ako body hrán z obrázku. Mapu postačujúcu pre tento prípad tvorí drátený model, v ktorom sa však pri projektovaní nezobrazujú prekryté hrany. Určovanie pozície robota pomocou sledovania segmentov aj bodov hrán je rozvinuté v kapitole 7.

Obidve analyzované možnosti pritom počítajú so znalosťou výšky kamery nad podlahou a jej sklonom k podlahe. Registrácia využíva prepočet nad referenčnou rovinou robota, pretože v takomto prípade stačí spárovať

s mapou vhodne zvolené dva body na snímku z kamery (Veta 2 v sekcii 2.4).

5.1 Hľadanie pozície hrubou silou

Hoci je algoritmus založený na prechádzaní všetkých možných párovaní je použiteľný pre malý počet bodov vďaka vhodnému orezaniu. Generovanie párovaní si možno predstaviť ako rekurziu, ktorá postupne generuje všetky možnosti prípustných párov. V jednej hĺbke berie postupne v úvahu párovanie, ktoré neobsahuje prvky predchádzajúcich vnorení a jeho zrovnanie má s aktuálnou mapou prípustného konfiguračného priestoru robota nenulový prienik. Touto podmienkou sa vyradí množstvo párov už v treťom kroku, pokiaľ nebol na začiatku konfiguračný priestor robota nijak vyhradený. Ak je na začiatku prípustná iba malá oblasť pre pozície robota, tak hodne párovaní vypadáva už v prvom kroku. To je dosť pozitívne, pretože inak by každý krok násobil pamäťové a hlavne časové réžie rádovo násobkom počtu primitív daného typu na obrázku s počtom primitív v mape, ktoré vstupujú do algoritmu. Ako výsledok sa zoberie párovanie, ktoré vzniklo v najhlbšej hladine rekurie a obsahuje teda najviac dvojíc. Poloha robota je určená prienikom prípustných oblastí párov z maximálneho párovania.

Zrovnanie s mapou

Zrovnanie s mapou v danom algoritme predstavuje prienik konfiguračného priestoru s oblasťou možných polôh robota pri zvolení párovania jedného bodu obrázku s jedným bodom mapy. Výstupom daného zrovnania je upravená oblasť možných polôh robota. Vstupom je konfiguračný priestor robota, jeden bod obrázku a korešpondujúci bod mapy. V našom prípade je výhodné brať ťažiská segmentov, pretože segmenty je možné ľahko klasifikovať už pri ich určovaní v obraze pomocou farby. Pri konštantnej výške kamery a konštantnom sklone kamery od podlahy je možné zrátať vzdialenosť kamery od bodu v scéne. Samozrejme, pokiaľ tento bod neleží na priamke obrázku predstavujúcu horizont rovín rovnobežných s podlahou. Pre naše účely však môžeme predpokladať, že sklon a uhol záberu kamery takúto situáciu nepripúšťajú, alebo jednoducho takéto body v tomto kroku nebrať v úvahu. Zrovnanie dvoch bodov týmto spôsobom vytvorí kružnicu možných polôh robota. Navyše je možné pre každú takúto polohu určiť natočenie, v ktorom by sa mal robot nachádzať, aby videl daný bod mapy na obrázku, tak ako ho vidí. V praxi však nemôžme byť tak presný, pretože vo

Algoritmus 2: Rekurzívna funkcia na nájdenie najlepšieho párovania

Data:
$cifs$ - ťažiská klasifikovaných (napr. špecifickej farby) segmentov detekovaných na obrázku,
wfs - ťažiská v mape odpovedajúce typu $cifs$
cs - konfiguračný priestor robota
Result:
$parovanie$ - najlepšie párovanie,
cs - oblasť polohy robota pri najlepšom párovaní bodov
begin
  $parovanie$ = prázdne párovanie;
  for $cif \in cifs$ **do**
    for $wf \in wfs$ **do**
      cs = prienik cs s oblasťami polôh robota pri $< cif, wf >$ (zrovnanie s mapou);
      if $!cs.empty()$ **then**
        $priebezne\_parovanie = paruj(cifs - cif, wfs - wf, cs)$;
        if $priebezne\_parovanie.size() > parovanie.size()$ **then**
          $parovanie = priebezne\_parovanie + < cif, wf >$;
  return $parovanie$
end

výpočtoch vzniká značná chyba. Daný problém je však možné vyriešiť definovaním chyby, ktorá rozširuje možné polohy robota vo všetkých smeroch. Výsledok je teda orezanie konfiguračného priestoru robota, tak aby bolo použiteľné v ďalšom kroku pri výbere ďalšieho páru. Návrh implementácie konfiguračného priestoru robota je uvedený v sekcii 5.3.

5.2 Párovanie na základe dvojíc párov

Aby malo párovanie zmysel, tak by body nemali ležať vo výške kamery, kde robot nie je schopný určiť ich relatívnu vzdialenosť a natočenie z polohy na snímku. Bod z obrázku kamery značím ako c_i, zatiaľ čo bod v mape ako w_i. Párovanie takýchto dvoch bodov je označené ako $< c_i, w_i >$.

Pre začiatok je dobré si uvedomiť, že nekorektne zvolené dvojice nemajú veľký vplyv na korektnosť výsledku.

Veta 3 *Ak sa vo dvojici vyskytuje rovnaké ťažisko z obrazu pri iných bodoch mapy, tak neexistuje možná pozícia robota v scéne.*

Dvojica $<< c, w_1 >< c, w_2 >>$ negeneruje žiaden rozumný prienik možných oblastí robota, pretože inak by musela byť vzdialenosť aj orientácia robota k w_1 rovnaká ako k w_2. Z čoho vyplýva, že prienik by existoval, iba pokiaľ by sa v pohľade do scény w_1 prekrývalo s w_2.

Veta 4 *Ak sa vo dvojici párov vyskytuje rovnaký bod mapy pri iných ťažiskách obrazu, tak neexistuje možná pozícia robota v scéne.*

Dvojica $<< c_1, w >< c_2, w >>$ negeneruje žiaden rozumný prienik možných oblastí robota. Pretože natočenie robota je závislé od $c_1.x$ resp. $c_2.x$ a vzdialenosť je závislá od bodu od $c_1.y$ resp. $c_2.y$. Oblasť možných pozícií, kde je vzdialenosť a natočenie rovnaké existuje pri rovnakom w iba ak by sa c_1 rovnalo c_2.

Na druhú stranu môže nastať situácia, keď sú ťažiská rôznych segmentov v rovnakom bode. Takáto dvojica nedáva jednu pozíciu, ale celú kružnicu ako iba pri jednom páre. Riešením je spracovávať takto iba jeden zo segmentov, ktoré majú na obraze ťažiská v jednom bode.

Vylepšenie algoritmu hrubej sily si možno predstaviť ako nájdenie prvého páru, ktorý spojí relatívny priestor robota s priestorom scény. A potom hľadanie natočenia určeného druhým možným párom. Pri tejto dvojici párov má robot vo väčšine prípadov jednu možnú pozíciu. Túto však treba brať s určitou rezervou vzhľadom k nepresnostiam v detekcii a k posunutiam ťažísk, viď sekciu 4.5. Pri tejto pozícii sa ďalej preberú ostatné možnosti párovania v relatívne rýchlejšom čase vďaka určitému predspracovaniu mapy.

Výhodou tohto algoritmu je okrem trocha lepšej časovej zložitosti to, že výstupom nie je iba jedno maximálne párovanie určujúce jednu pozíciu robota, ale halda dvojíc párovaní. Najlepšia pozícia tak odpovedá vrcholu haldy. Výpočet však naďalej zostáva príliš pomalý na to, aby bežal v reálnom čase pri pohybe robota. Avšak ide zaručiť, aby algoritmus najprv spracovával párovania, ktoré generujú lepšie pozície a tým udržovať na vrchole haldy vhodnú pozíciu i počas výpočtu. Časová zložitosť je $O(n^3.m.s(m).k(m))$, kde n je počet detekovaných ťažísk na obrázku ($n = |cifs|$) , m je počet možných viditeľných ťažísk v mape ($m = |wfs|$), $s(m)$ je maximálny počet susedov bodu mapy a $k(m)$ je zložitosť vyhľadania vhodných pevných párov pri pevnej pozícii robota. Pamäť použitá na haldu nepresahuje počet možných dvojíc párov, ktorý je $O(n^2.m^2)$. Pre rýchle

Algoritmus 3: Párovanie na základe dvojíc párov

Data:
$cifs$ - ťažiská klasifikovaných segmentov detekovaných na obrázku,
wfs - ťažiská v mape odpovedajúce typu $cifs$

Result:
H - halda pozícií z dvojíc párov s hodnotou, ktorá určuje koľko párov sa v danej pozícii robota podarilo spárovať. Halda je usporiadaná podľa danej hodnoty, tak že jej koreňom je maximálna takáto hodnota.

begin

for $c_1 \in cifs; w_1 \in wfs$ **do**
  for $c_2 \in cifs : c_2 > c_1$ **do**
    for $w_2 : dist_{ref}(c_1, c_2) \approx dist_{ref}(w_1, w_2)$ **do**
      if $\exists P = pozicia_{robota}(< c_1, w_1 >, < c_2, w_2 >)$ **then**
        $pocet\_parov = 2$;
        for $c \in cifs : c > c_2$ **do**
          if $\exists w \in wfs :< c, w >$ *pripúšťa pozíciu* P **then**
            $pocet\_parov + +$;
        $H.push(< pocet\_parov, P >)$;
  return H

end

vyhľadanie susedných bodov mapy, ktoré zabezpečujú vhodné natočenie relatívneho priestoru robota (w_2) je vhodné mapu predspracovať tak, aby v každom jej bode boli odkazy na ostatné body, ktoré je možné spolu s týmto bodom vidieť v nejakom prípustnom pohľade z kamery. A tieto odkazy usporiadať podľa vzdialenosti od daného bodu. Maximálny počet odkazov z jedného bodu, tak odpovedá $s(m)$. A vyhľadanie pevných párovaní k danej dvojici párov je možné realizovať prienikom výberu vhodne vzdialených bodov v správnom polpriestore. Polpriestor je určený rovinou ohniska kamery a bodu v mape daných dvoch párov. Smeruje buď nahor, pokiaľ párujem ťažisko nad priamkou c_1, c_2, alebo nadol v opačnom prípade.

5.3 Optimalizácie využitím konfiguračného priestoru robota

Jediný účinný spôsob ako znížiť zložitosť kombinovania párov medzi mapou a obrázkom je zohľadniť ich priestorové rozloženie. To je úzko zviazané s pozíciou kamery pri pohľade do scény a teda aj so samotnou pozíciou robota (kapitola 2). V tejto sekcii sú rozobrané možnosti ako rozčleniť priestor možných pozícií robota. Stačí totiž približné určenie prípustnej oblasti kamery a testovať, či poloha pri nejakom párovaní leží v nejakej takejto oblasti.

Konfiguračný priestor robota je teda priestor všetkých možných jeho pozícií. Pozícia je určená jeho 2D súradnicami na podlahe v mape a natočením vo vodorovnej rovine podlahy.

Pri každom spracovaní snímku a určovaní pozície sa tak pracuje nad množinou oblastí možných polôh v tomto priestore. Jednotlivé body alebo oblasti môžu mať pravdepodobnostné ohodnotenie, ktoré však pre samotný algoritmus nie je podmienkou. Naopak je omnoho efektívnejšie v algoritmoch 2 a 3 pracovať s ohodnotením binárnym, kde je pozícia robota buď neprípustná alebo prípustná. Z pravdepodobnostného ohodnotenia to možno dostať správnym zvolením prahu prístupnosti pozícií.

Hlavnou myšlienkou registrácie bodov mapy s bodmi v obraze je určenie vzdialenosti k danému bodu na základe jeho výšky (z-ovej súradnice) v mape. Tento výpočet však nemá zmysel pokiaľ je bod vo výške kamery. Danú krajnú možnosť výsledku je však možno eliminovať tým, že sa zvolí vhodný sklon kamery k podlahe. Kamera tak vidí na podlahe iba do určitej vzdialenosti a nie body na horizonte. Body blízke výške horizontu nesú informáciu iba o natočení kamery vzhľadom k nim, zatiaľ čo ostatné body po spárovaní vravia, že sa nachádzame v nejakom ich okolí. Informácia o vzdialenosti robota k daným bodom je teda presnejšia, čím sú body nižšie (vyššie) od výšky kamery. Algoritmus vytvára pri danom zrovnaní pre dvojicu bodov (mapa - obrázok) v referenčnej rovine akési medzikružie zo stredom pod spracovaným bodom mapy. Rozdiel medzi vnútorným a vonkajším polomerom je závislý na presnosti informácie. Čím je bod bližsie a nižšie (resp. vyššie) od výšky kamery, tým je medzikružie užšie. Bližšie popísanie vzťahu medzi projektovaním bodu do referenčnej roviny a jeho rozptylu a váhy je popísané v sekcii 3.2. Do šírky daného medzikružia je vhodné započítať aj predpokladaný posun ťažiska, viď. sekciu 4.5.

Pri výpočte je veľmi dôležité zistiť, či dané medzikružie má nejaký spo-

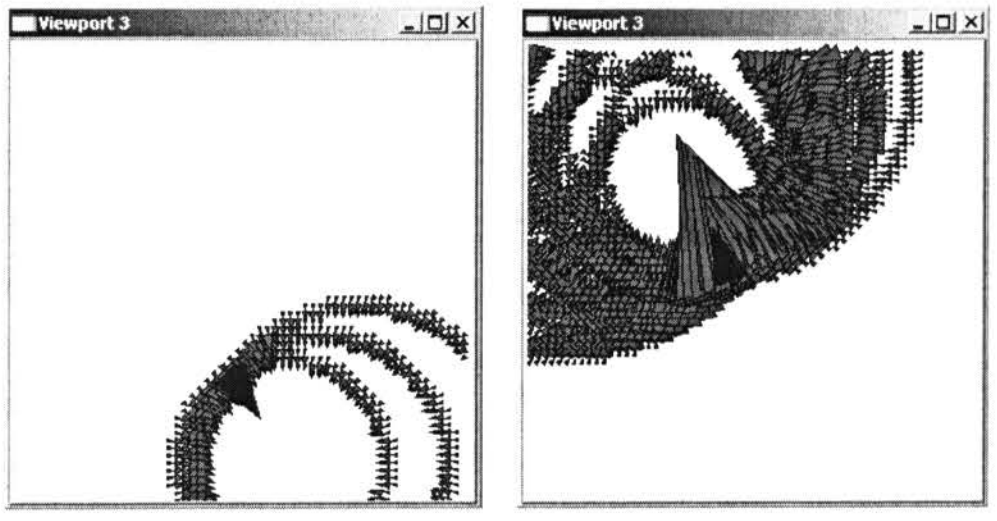
ločný prienik s možnými pozíciami robota. Pokiaľ áno, tak daný prienik spočítať a vyhlásiť za nové možné pozície robota. Keďže je algoritmus lokalizácie realizovaný hrubou silou je tento výpočet veľmi častý a vplýva tak veľmi na rýchlosť celej lokalizácie. Je preto dôležité správne zvoliť reprezentáciu možných polôh robota, tak aby bol dotaz na prienik dostatočne rýchly.

Implementácia pomocou bitmapy

Najintuitívnejšie je zobrazenie možných polôh do pola, ktoré reprezentuje navzorkovaný priestor mapy. Stačí použiť dvoj-rozmerné pole, ktorého prvkami bude natočenie - v prípade možnej pozície alebo iná zvolená hodnota v prípade neprístupnosti. Nutné je doplniť rozmedzie natočenia, či už globálneho alebo lokálneho charakteru.

Vykresľovanie do takejto reprezentácie hodne pripomína vykresľovanie do bitmapového obrázku. Na zafarbenie medzikružia je možné použiť upravený Bresenhamov algoritmus.

Obrázok 5.1: Konfiguračný priestor implementovaný bitmapou



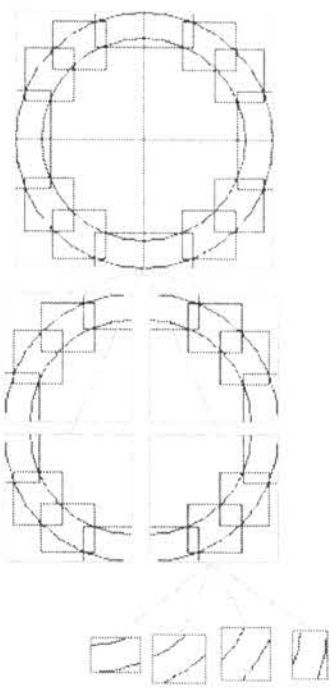
Na obrázku je vidieť zjednotenie váh ohodnotených pozícií pri správnom párovaní.

Implementácia pomocou stromu obálok

Obálkou pre jednotlivé oblasti sú ortogonálne kvádre, ktoré sú ďalej spájané do väčších obálok atď. Obálky sa môžu navzájom prekrývať. Takto vzniká strom. Pri dotaze na nejakú ortogonálnu oblasť sa v uzle zistia potomkovia, ktorý majú s danou oblasťou prienik a dotaz sa analogicky aplikuje na obálky potomkov až pokiaľ sa nedostane k listom. Najmenšie obálky v listoch sa volia buď tak malé, aby pri prienikoch nebolo treba obálky

zmenšovať, alebo sa pri prienikoch obálky zmenšujú. Dotaz na existenciu prieniku medzi dvoma kvádrami tak pozostáva v šiestich jednoduchých porovnaní. Dotaz prieniku obálky s medzikružím popisuje algoritmus 4.

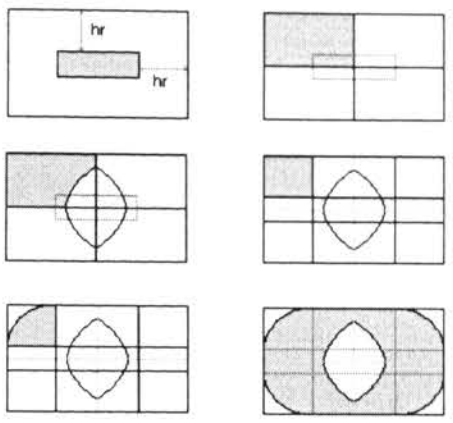
Obrázok 5.2: Rozdelenie medzikružia do obálok



Pokiaľ obálka nemá prienik tak, zapíše do daného uzlu príznak, ktorý označí jeho podstrom za neplatný. A v strome tak nakoniec zostanú iba obálky, ktoré majú prienik v danom dotaze.V ďalšom postupe lokalizačného algoritmu je dôležité, aby sa po odskúšaní daného párovania situácia navrátila do pôvodného stavu, čo je realizované jednoducho pomocou spomenutých príznakov platnosti. To by však nebolo také jednoduché, ak by sa obálky v listoch orezávali. V tom prípade by sa museli uchovávať informácie o zmene obálok, čo by znamenalo ďalšie časové a hlavne pamäťové réžie.

Mohlo by sa zdať, že pokiaľ bude výsledkom nejaká malá množina obálok, tak bude pozícia robota ešte stále dosť nepresná. Čo však nie je pravda, pretože hlavným výstupom algoritmu je párovanie s mapou. Na základe ktorého sa ďalej môže analiticky spočítať pozícia veľmi presne. A to napríklad metódou najmenších štvorcov uvedená v sekcii 3.1.

Obrázok 5.3: Prienik obdĺžnika s medzikružím



Algoritmus 4: prienik obdĺžnikovej obálky s medzikružím

Data:
hr - vonkajší polomer medzikružia
lr - vnútorný polomer medzikružia
C - stred medzikružia
UL, UR, DR, DL - vrcholy obálky
KA, KB, KC, KD - uniformné vertikálne rozdelenie rozšírenej obálky o hr, lr na rovnaké štyri disjuktné kvadranty
Result:
obálka má/nemá prienik s medzikružím
begin
  if *C nie je v obálke rozšírenej z každej strany o hr* **then**
    return *FALSE*
  BÚNO: C je v kvadrante KA [obrázok 5.3 b - vpravo hore];
  if *C nie je ďalej než lr od DR [obrázok 5.3 c]* **then**
    return *FALSE*
  if *C je v časti α a C nie je bližšie než hr od UL [obrázok 5.3 e]* **then**
    return *FALSE*
  return *TRUE*
end

Kapitola 6

3D mapa

Mapa uchováva informácie o jednotlivých objektoch v priestore okolo robota. Objekty mapy sú jednotlivé primitíva, ktoré sa dajú dobre zrovnávať v obrázku. Ako sú napríklad ťažiská segmentov, významné hrany, kostry segmentov alebo aj jednotlivé elementárne plochy.

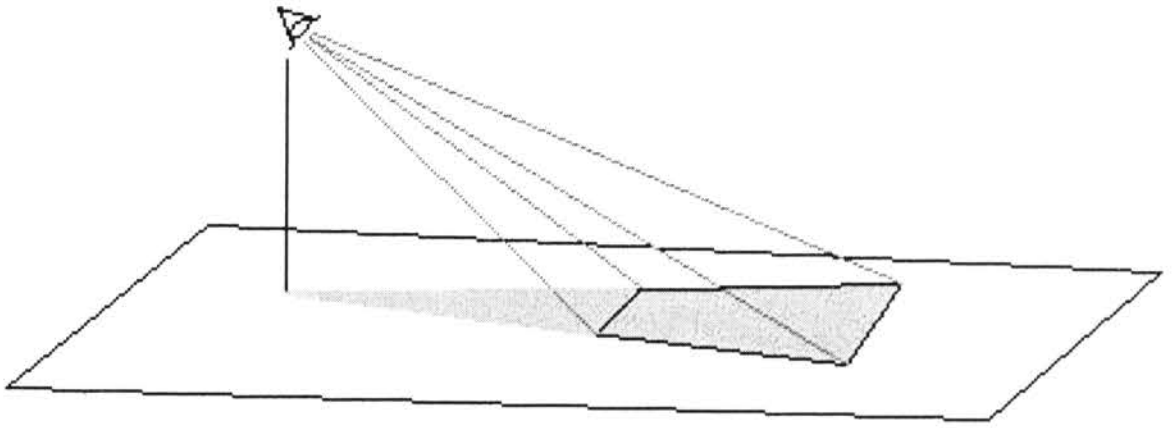
Univerzálnu mapu si je možné predstaviť ako elementy uložené na pozícii svojho ťažiska. Elementy sú trojuholníkové plôšky a okrem svojich troch vrcholov obsahujú odkaz na histogram, poprípade iné príznaky, ktoré charakterizujú segment daného elementu na obrázku z kamery. Vhodné je, aby element obsahoval ešte ďalšie doplnkové redundantné informácie, ktoré slúžia na urýchlenie výpočtov. Zobrazovanie elementu je vždy určené podľa orientácie jeho plochy, či smeruje od kamery alebo k nej. Hoc je danú orientáciu možné určiť už poradím vrcholov, je zobrazovanie rýchlejšie ak je normála uložená v mape. Požiadavka na trojuholníkový tvar však spôsobí, že sa plochy rozpadnú na niekoľko trojuholníkov. Preto je potrebné držať v každom takom trojuholníku index segmentov, ktorých trojuholníkové elementy sú navzájom spojené. Príznakový vektor tak charakterizuje tieto segmenty a nie ich elementárne trojuholníkové plôšky. Ak môže nastávať v mape prekrývanie objektov, tak je potreba to zohľadniť pri zobrazovaní segmentov v konkrétnom pohľade. Pritom platí, že ak je segment len čiastočne prekrytý iným segmentom, tak nebude navrátený a ďalej spracovávaný. Zobrazovanie s prekrývaním je možné implementovať ľahko pomocou Z-bufferu, kde stačí si jednotlivé elementárne plôšky z určitého výrezu mapy utriediť podľa vzdialenosti od ohniska kamery. Vykresľujú sa odzadu. Takže v konečnom dôsledku sú prekryté vzdialenejšie, nahradené plôškami, ktoré

ich prekrývajú. Existuje však situácia, keď majú dva segmenty ťažiská v opačnom poradí než v tom, v ktorom sa prekrývajú. Tu stačí elementy rozrezať podľa roviny druhého elementu. A vzniknutý štvoruholník je treba rozdeliť na trojuholníky, aby sa zachovala trojuholníková povaha mapy. To, že daná operácia postačuje plynie z tvrdenia, že rovina nebude prekrývať rovinnú plôšku, ktorá neleží pod ňou.

Pre algoritmus 3 je vhodné k jednotlivým segmentom ukladať aj zoznam ostatných segmentov rovnakého príznakového vektoru utriedený podľa vzdialenosti k nemu.

Oknom pohľadu z jednej pozície je ihlan od ohniska kamery v smere pohľadu. Jeho prienikom na podlahe je štvoruholník. Avšak môžu byť viditeľné aj body, ktoré možno projektovať na podlahu medzi projektovaným bodom kamery a štvoruholníkom. Oblasť, nad ktorou je možné vidieť objekty v mape je teda určená trojuholníkom, pokiaľ kamera nevidí horizont.

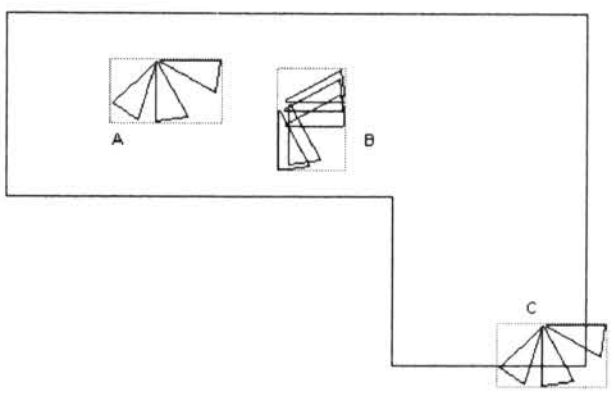
Obrázok 6.1: Pohľad do mapy



Dotaz do mapy je smerovaný aj na základe oblastí, kde by sa robot mohol nachádzať. Môže to byť celý priestor, ale väčšinou je požiadavka smerovaná na základe vopred známej množiny možných pozícií robota. V tomto prípade sa s prekrývaním segmentov nepracuje a navrátia sa všetky segmenty v danom zjednotení pohľadov. Pre zjednodušenie je lepšie používať dotaz na obdĺžnik nad rovinou podlahy, ktorý má v porovnaní s inými geometrickými štruktúrami menšie réžie v spracovaní. Mapu je možné prezentovať trojrozmernými priestorovými štruktúrami. Pre prehľadnosť a jednoduchosť dotazov som sa rozhodol používať dvojrozmerné štruktúry, ktoré predstavujú projekciu objektov do roviny podlahy. Navyše som upustil od zjednotenia jednotlivých trojuholníkov reprezentujúcich záber pohľadu a pracujem iba s ich obálkami. Dotaz môže byť formovaný aj pomocou

množiny viacerých obálok, ktoré pokrývajú projekciu primitív viditeľných zo skupiny možných polôh kamery. Spájanie jednotlivých záberov pohľadu je pri výbere obdĺžnikovej obálky riešené pomocou min-max algoritmu. Najmenšia x-ová a y-ová súradnica určuje ľavý predný bod okna a maximálne hodnoty určujú pravý zadný bod okna. Pri spájaní je možné brať v zreteľ na disjuktné zábery pohľadu a algoritmus inkrementálnym priberaním záberov bude spájať do jedného okna iba tie, ktorých okná by sa prekrývali. Bude tak udržovať zoznam disjuktných obdĺžnikov, ktoré však môžu byť spájané v ďalšom kroku dohromady. Pri pribraní nového záberu tak treba prejsť všetky doposiaľ vytvorené okná a zistiť, s ktorými má nové okno pre daný záber prienik. Z týchto vyčlenených obdĺžnikov vznikne nakoniec jeden zjednocujúci obdĺžnik, ktorý ich nahradí.

Obrázok 6.2: Obálky pohľadov do mapy



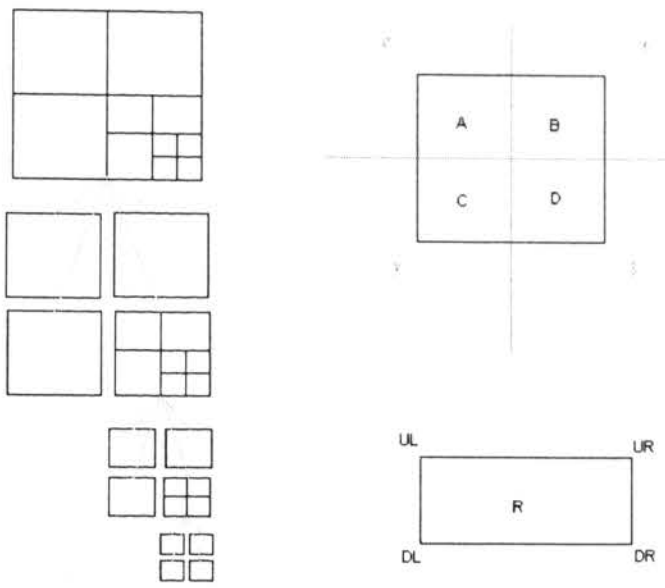
Nevýhodou takéhoto prístupu je, že budú navrátené primitíva, ktoré nie je možné vidieť zo žiadnej východzej pozície robota. Takéto body budú zbytočne zaťažovať nasledujúci výpočet lokalizácie. Tieto body je však možné eliminovať testovaním, či ležia v nejakom pohľade z daných pozícií robota.

6.1 Quadtree

Je dátová štruktúra, ktorá slúži na vyhľadávanie 2D priestorových dát. Je to strom, ktorého každý uzol má štyroch potomkov. Uzol reprezentuje štvorec v priestore. Dáta uložené pod daným uzlom ležia práve v tomto štvorci. Potomkovia majú túto plochu rozdelenú disjuktne na štyri menšie podštvorce, ktorých zjednotenie je práve štvorec ich rodiča.

Dotaz je realizovaný od koreňa stromu až po listy, ktoré majú z vyhľadávaným oknom nejaký prienik. Predpokladom vstupu do uzlu je, že vyhľadávané okno má s príslušnou plochou daného uzlu nenulový prienik. To je automaticky zaručené aj pri vstupe do potomkov.

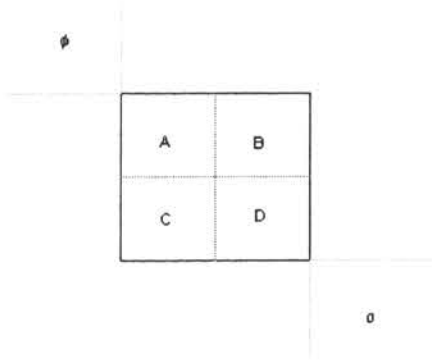
Obrázok 6.3: QuadTree



Treba brať ohľad na primitíva, ktoré zaberajú viac ako jeden uzol stromu. Buď sa budú v návratovom zozname vyskytovať násobne, alebo ich treba prispôsobiť tak, aby mali príznak spracovania. Pri ukončení vyhľadávania sa tento príznak navráti do pôvodnej hodnoty. To je realizované prechodom výsledného zoznamu odkazov na primitíva uložené v danom strome.

Vylepšením danej štruktúry by mohlo byť ukladanie zoznamu primitív už do samotných uzlov stromu a nielen do listov. Pamäť na odkazy pre primitíva takto síce narastie priamo úmerne s hĺbkou stromu, ale výpočet dotazu sa značne urýchli. A to hlavne pri veľkých vyhľadávaných oknách. Pretože pokiaľ dotaz prekryje celú plochu uzlu, tak sa navráti priamo celý zoznam v danom uzle a jeho potomkov už nie je treba spracovávať.

Obrázok 6.4: Rozdelenie uzlu QuadTree



6.2 BSP strom

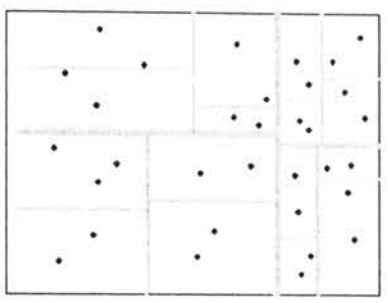
Vychádza s analógie nad vyhľadávaním v jednorozmernom poli. Obecne je možné deliť rovinu ľubovolnou priamkou. No pre zjednodušenie nám postačí

Algoritmus 5: prienik obdĺžnikovej obálky s Quadtree

Data:
node - uzol stromu Quadtree
R - obálka, ktorá má a plochou reprezentovanou stromom s koreňom *node* nenulový prienik
Result:
res - dáta vo vnútri obálky R
begin
  if $UL \in \psi \&\& DR \in \omega$ **then**
    $res.add(node.data)$;
    Return;
  if *node je listom* **then**
    for $d \in node.data$ **do**
      if $d \in R$ **then**
        $res.add(d)$;
  else
    if $R.UL \in \alpha$ **then**
      $intersection(node.A, R, res)$;
    if $R.UR \in \beta$ **then**
      $intersection(node.B, R, res)$;
    if $R.DL \in \gamma$ **then**
      $intersection(node.C, R, res)$;
    if $R.DR \in \delta$ **then**
      $intersection(node.D, R, res)$;
end

každou párnou úrovňou stromu rozdeliť plochu rodiča pozdĺž osi x a každou nepárnou úrovňou stromu rozdeliť plochu pozdĺž osi y. Poloha deliacich čiar je určená mediánom prvkov, ktoré zahrňuje daný uzol. Každý uzol tak má dvoch potomkov. Pri dotaze treba rovnako zistiť, či vyhľadávané okno má s potomkom nejaký prienik. Pokiaľ áno, tak následne dotaz rovnako rekurzívne aplikovať. Zostáva taktiež zachovaná podmienka na neprázdny prienik s uzlom, čo trochu zjednodušuje výpočet. Výhoda tejto štruktúry je hlavne v mapách, kde nastávajú zhluky príznakov. V predchádzajúcom quadtree by tak mohol nastávať prípad, keď je v jednom uzle uložený veľký počet primitív, zatiaľ čo susedné uzly sú takmer prázdne. To má veľký vplyv na hĺbku stromu alebo na spracovanie daného listu.

Obrázok 6.5: BSPTree



Ďalšie možnosti reprezentácie mapy sú napríklad pomocou R-Tree, ktoré je vhodné najmä pre zložitejšie objekty ako samotné body. Vďaka flexibilnosti obálok, ktoré sa môžu navzájom prekrývať umožňuje ukladať priestorové dáta bez zbytočnej redundancie, čo šetrí pamäť pre štruktúru a neraz aj časové nároky pre dotaz.

Kapitola 7

Trackovanie pohybu

Pri pohybe robota scénou je možné využiť informácie o predchádzajúcej pozície a o relatívnej prejdenej vzdialenosti. Vhodným senzorom sú napríklad enkodéry, ktoré merajú otočenia kolies. Z odometrie sa potom dá zrátať približný posun robota od poslednej zaznamenanej pozície. Konečná poloha je vyjadrená ako súčet týchto posunov. Bez ďalších informácií sa ale po určitej vzdialenosti nemusí vôbec stotožňovať s reálnou pozíciou robota v scéne. Dôvodom je inkrementálna chyba, ktorá k sebe pri každom kroku pripočítava elementárne chyby z enkodérov.

Cieľom trakovania pohybu v mape je upresnenie polohy získanej z enkodérov, resp. ukázať, že pri vhodných podmienkach je možné udržiavať polohu iba pomocou obrazu z kamery.

V sekciách 7.1 a 7.2 sú popísané algoritmy na zistenie posunu, ktorý vyjadruje korekciu pozícií segmentov (resp. hranových bodov) medzi snímkom a pohľade do mapy z predchádzajúcej pozície. Treba poznamenať, že posun segmentov je realizovaný zjednodušením cez posun ich ťažísk na snímku ako je uvedené v sekcii 4.5. Výhodou takýchto postupov je, že výsledkom sú už párovania medzi mapou a snímkom. Pri ich dostatočnom počte je tak možné vyjadriť presnejšiu polohu robota výpočtom zo sekcie 3.1. Vyjadrenie pohybu naďalej zostáva z princípu relatívne, pretože je závislé od predchádzajúcej polohy. Navyše aj tu vzniká v každom kroku elementárna chyba, ktorá môže spôsobiť v niektorých situáciách úplné stratenie absolútnej pozície v mape.

Inou možnosťou ako robustne trakovať pozíciu by bola napríklad Monte Carlo lokalizácia, ktorá sleduje veľké množstvo možných trajektórií pohybu

s určitými váhami. Nie je pritom nutné si pamätať celú trajektóriu, ale stačí pracovať iba s jej posledným bodom, ktorý predstavuje jednu vzorku pozície v aktuálnom časovom okamžiku. Pri tejto možnosti trakovania je síce možné použiť algoritmy na upresnenie pozície z obrazu na každej vzorke (ako je uvedené v [8, Cai, 2006]), ale pri veľkom počte vzoriek by bola metóda veľmi časovo náročná. A tak je výhodnejšie posúvať vzorky pozície iba podľa schválne zašumených dát z enkodérov. Váhy jednotlivých vzoriek je možné určiť podľa bodov obrázku, ktoré by mali vyhovovať náhodne vybraným bodom na viditeľných segmentoch pri pohľade z pozície vzorky.

7.1 Mean shift

Je algoritmus na určenie posunu segmentu v lokálnom okne zo spätnej projekcie jeho histogramu, viď sekciu 4.4. Ako napovedá názov výsledkom je posun, ktorý koriguje odhad polohy segmentu na obrázku z kamery. V našom prípade je použitý na upresnenie polohy robota. Pri predpokladanej pozícii sa vyberú z mapy výrazné viditeľné segmenty. Projektovaná poloha každého takéhoto segmentu na obrázku je algoritmom Mean Shift upravená tak, aby odpovedala pri krátkom posunutí čo najviac snímku z kamery. Vzniká automaticky párovanie medzi segmentami v snímku a segmentami v mape. A pomocou výpočtu v sekcii 3.1 nakoniec máme presnejšiu pozíciu robota v scéne.

Nech I_{H_f} je spätná projekcia histogramu H_f do obrázku z kamery. Histogram H_f charakterizuje segmenty špecifickej farby typu f pri rôznych možných pohľadoch z kamery. Je uložený ako súčasť mapy. Pri vytváraní mapy ho je možné vytvoriť zo vzoriek častí obrázkov scény obsahujúcich iba daný typ segmentov. V poli I_{H_f} tak má každý bod hodnotu odpovedajúcu hodnote danej farby v histograme H_f. Zabezpečí sa tak viditeľnosť iba pre daný typ segmentov.

Korekcia pozície je na I_{H_f} realizovaná vďaka vhodne zvolenému lokálnemu oknu W_L. V danom okne sa zráta vážený priemer súradníc bodov podľa váh určených hodnotami v I_{H_f}. Vzniká tak nový stred okna so súradnicami $(1/\sum_{(X,Y)\in W_L} I_{H_f}(X,Y))\cdot(\sum_{(X,Y)\in W_L} X\cdot I_{H_f}(X,Y), \sum_{(X,Y)\in W_L} Y\cdot I_{H_f}(X,Y))$. Postup je možné iterovať a zabezpečiť tak situáciu, keď časť segmentu leží mimo lokálneho okna.

Lokálne okno W_L, na ktorom sa ráta vážený priemer súradníc sa určí rozšírením zjednodušenej projektovanej obálky skúmaného segmentu W_S. Váhu korekcie polohy segmentu je možné definovať zo súčtu váh v rozdiele

posunutých okien $W'_L \setminus W'_S$. Vychádzam pritom z predstavy, že v ideálnom prípade sú váhy sústredené v okne tesne ohraničujúcom segment W'_S, zatiaľ čo v najhoršom prípade sa môže stať, že algoritmus zostane na hrane medzi segmentami. Pri takejto situácii, by však musel byť rovnaký súčet váh v každom smere. S reálnymi váhami je však veľmi malá pravdepodobnosť, že by sa algoritmus hneď pri prvej iterácii ocitol v takejto situácii. Ďaleko častejšie nastáva situácia, keď sú váhy v rozšírenom okne skoro všade rovnaké. V tomto prípade taktiež algoritmus nedáva žiadnu informáciu, pretože nevie rozoznať farebný objekt na pozadí rovnakej farby, alebo daný segment nemá s oknom W_L žiaden prienik. Váhy je treba nastaviť tak, aby mali v týchto situáciách nulovú hodnotu, zatiaľ čo v spomínanej ideálnej situácii hodnotu 1. Rovnica pre váhu by tak mohla vyzerať napríklad nasledovne. Orezaná tak, aby jej hodnoty boli nezáporné.

$$\frac{|W_L| - |W_L|}{|W_L|} - \frac{\sum_{a_i \in W_L} I_{H_f}(a_i) - \sum_{a_j \in W_S} I_{H_f}(a_j)}{\sum_{a_i \in W_L} I_{H_f}(a_i)}$$

kde $|W|$ je počet bodov v okne W.

Dané váhy sa ďalej použijú pri výpočte pozície robota z takto korigovaných segmentov obrázku spárovaných zo segmentami mapy, z ktorých boli v jednotlivých prípadoch brané histogrami a obálky. Chybou takto získanej pozície je chyba výpočtu pozície robota definovaná v sekcii 3.2.

7.2 ICP (Iterated Closest Point)

V množstve prípadov však okolie robota nemusí byť dobre prispôsobené na predchádzajúcu metódu. Mean shift pracuje dobre na malých lokálnych jednofarebných plôškach, ktoré ležia v prostredí inej farby. V mnohých prípadoch nie sú v jednom pohľade viditeľné celé takéto plôšky, alebo sa prekrývajú s rovnako farebnými plôškami. Tu je segmentácia pre nedostatok vhodných segmentov nepoužiteľná a je nutné použiť iné možnosti trakovania. Jednou takouto možnosťou je zamerať sa na hrany. Hrany je možné spracovávať vektorizáciou a ďalej vo vektorovej podobe. Druhou možnosťou je hrany spracovávať ako binárny obrázok, ktorý sa zrovnáva s projekciou bodov pri pohľade do mapy. Jedným z algoritmov, ktoré zrovnávajú dve množiny bodov je ICP algoritmus. Tento algoritmus funguje iba pokiaľ je známa približná poloha kamery v mape.

Vstupom algoritmu sú dve množiny bodov - body modelu $\{m_i\}_{i=1}^{N_m}$ a body dát $\{d_i\}_{i=1}^{N_d}$. V našom prípade je modelom scéna, poprípade jej časť,

Algoritmus 6: ICP algoritmus

Data:
Img - obrázok z kamery
$pType$ - typ bodov, na základe ktorých prebieha registrácia
Map - mapa scény
a - približná poloha robota
Result:
presnejšia poloha robota
begin
  P_{img} = projekcia bodov typu $pType$ z Img do referenčnej roviny
  $Nearest_{img}$ = pole odkazov na najbližšie dané body P_{img}
  while *true* **do**
    P_{scn} = projekcia bodov typu $pType$ z Map do referenčnej roviny pri polohe a
    ϕ = párovanie P_{img} s P_{scn} určené polom $Nearest_{img}$
    if *párovanie sa v predchadzajúcom kroku nezmenilo* **then**
      return a
    a = poloha pri lepšom párovaní ϕ
end

v ktorej sa robot pohybuje. Dáta sú namerané príznakové body z kamery. Algoritmus v [3, Fitzgibbon, 2001] je navrhnutý pre senzory ktoré navrátia dáta už v 3D podobe (napr. laserový scaner). Pre naše účely sa však uspokojím s dátami, ktoré budú odpovedať bodom z obrázku umiestneným do referenčnej roviny.

V prvom kroku je nutné nájsť zobrazenie $\phi(i)$, ktoré mapuje bod dát d_i do bodu mapy $m_{\phi(i)}$. Obvykle sa volí výberom najbližšieho bodu mapy odpovedajúcemu bodu d_i. Priestor je možné uniformne navzorkovať a v každom bode mriežky predpočítať vzdialenosť a odkaz k najbližšiemu bodu. Vďaka záplavovému (Dijkstrovmu) algoritmu je možné vytvoriť mriežku v lineárnom čase (v závislosti na jej počtu bodov). Takúto vzdialenostnú mapu je možné spočítať globálne cez celý model alebo pre aktuálne namerané dáta pri konkrétnom prípade registrácii. V druhom prípade je mriežka 2D počíta sa iba na referenčnej rovine alebo na samotnom snímku snímku. Na rozdiel od [3, Fitzgibbon, 2001] sa budem zaoberať touto druhou možnosťou, pretože nebudem transformovať body dát do modelu, ale body modelu do snímku. Vyplýva to s toho, že dáta nemajú 3D polohu a ich zobrazením v scéne by neboli body ale priamky. Zatiaľ, čo pri projektovaní modelu do snímku (resp. referenčnej roviny rovnobežnej s podlahou)

je situácia jednoduchšia, a navyše iba v 2D.

Na výpočet polohy pri danom párovaní sa použije metóda najmenších štvorcov popísaná v sekcii 3.1.

Trackovanie za pomoci stereovision

Určiť relatívny pohyb robota po scéne pri jednej kamere pokiaľ nie je známa jeho približná pozícia v mape je všeobecne bez priebežnej rekonštrukcie mapy takmer neriešiteľná úloha. Avšak v prípade, že máme aspoň dve skalibrované, na robotovi pevne umiestnené kamery, tak je doplnená chýbajúca informácia o priestorovom hĺbkovom rozložení bodov. Vďaka tejto čiastočnej znalosti polohy jednotlivých rozpoznaných príznakov na obrázkoch je možné robota trakovať takmer rovnakými metódami ako pri známej približnej polohe v 3D mape a jednej kamere.

Napríklad O.Faugeras [2] sa pokúša sledovať pohyb robota pomocou stereo-vision tak, že zrovnáva priebežne generované 3D mapy úsečiek. Tieto mapy sú vytvorené v jednom kroku vektorizáciou hrán na oboch obrázkoch a ich následnou registráciou.

7.3 Monte Carlo lokalizácia spárovaných segmentov

Pri trakovaní pozície pomocou MeanShift algoritmu bola zaznamenaná určitá neistota. Táto neistota bola vyjadrená pomocou váh, ktoré boli využité pri zarátaní jeho posunutej pozície do pozície robota. Chybu, ktorá mohla takto nastať, je však možné filtrovať omnoho robustnejšie. Myšlienkou filtrácie je robustne sledovať jednotlivé polohy segmentov v relatívnom priestore robota, podobne ako u Y.Cai,2006 [8]. Na situáciu je tak možné nazrieť ako na pohyb segmentov a nie pohyb robota v scéne. Vychádza sa z predpokladu, že na začiatku je dobre zvolené párovanie segmentov medzi obrázkom a scénou. Pri jednotlivých krokoch sa sledujú iba pohyby jednotlivých segmentov. Každý segment sa sleduje nezávisle na ostatných až do momentu, keď je nutné aktualizovať pohľad do mapy a pribrať nové párovania segmentov.

Sledovanie jedného segmentu pomocou Monte Carlo lokalizácie prebieha v relatívnom priestore robota, viď sekciu 2.3. Zobrazenie medzi bodmi na obraze a bodmi v priestore robota teda musí byť známe. V ideálnom prípade

dokonca zostáva konštantné. To nastáva práve v situácii pri rovnej podlahe a nemeniacej sa polohe ani sklonu kamery vzhľadom k robotovi.

Filtrovanie spočíva v sledovaní viacerých možných trajektórií pohybu segmentu. Predpokladaný pohyb segmentov v priestore robota medzi jednotlivými krokmi je možné určiť rovnako, ako predpokladaný pohyb robota. S tým rozdielom, že bude záporný, pretože robot sa pohybuje proti segmentom. Predpokladaný pohyb robota je najjednoduchšie a najvhodnejšie určiť z enkodérov. Pokiaľ nie je takáto možnosť, tak je ho možné určiť na základe odometrie a predpokladanej reakcii motorov na príkazy plánovacej vrstvy robota. Poprípade je možné túto prvú predikciu posunu vzorky vynechať a riadiť sa iba predikciou z MeanShift. Pri použití takýchto relatívnych informácií treba vždy zarátavať určitú chybu. Tá je reprezentovaná posunom vzorky náhodne tak, aby nakoniec neležala presne na pozícii z enkodérov, ale v jej okolí. Striktne by mal byť nový model pozície určený započítaním pravdepodobnostného modelu rozloženia $p(x'_t|x_{t-1})$, ktoré určuje model zvolenej predikcie v kroku t.

$$p(x'_t|y_{0:t-1}) = \int p(x'_t|x_{t-1})p(x_{t-1}|y_{0:t-1})dx_{t-1}$$

Pre takto "zašumené"jednotlivé posuny vzoriek sa aplikuje ďalšia predikcia, ktorá posúva vzorku pomocou metódy MeanShift. Tu treba určiť pravdepodobnostný model posunutia $p(x_t|x'_t)$ tejto metódy, napríklad pomocou gausovského rozloženia. Ak sledujeme vzorky v referenčnej rovine, tak x-ový posun z MeanShiftu odpovedá iba x-ovému posunu v relatívnom priestore robota a rovnako to platí aj pre os y. Navyše si treba uvedomiť, že vzorky neobsahujú orientáciu, čím je ich konfiguračný priestor jednoduchší, ako v prípade, keď by sme chceli takto sledovať priamo pozíciu robota. Nezávislosť oboch súradníc nám preto povoľuje definovať rozloženie pomocou dvoch rozptylov na základe ktorých je možné vzorkovať súradnice oddelene. Takéto rozptyly je možné definovať rôznymi spôsobmi. Na ich odhad je možné napríklad použiť váhy a body z rozdielu medzi vonkajším W_L a vnútorným oknom W_S, ktoré sú definované v sekcii 7.1.

Predikcia pomocou MeanShift-u je tvaru:

$$p(x_t|y_{0:t-1}) = \int p(x_t|x'_t)p(x'_t|y_{0:t-1})dx'_t$$

Aby takto určená vzorka v čase t spĺňala funkciu filtrácie, tak jej musí byť priradená váha. Váhu je vhodné definovať z farebnej zhody takto posunutého segmentu mapy s farbou jeho novej oblasti na obrázku z kamery.

Na zrovnanie takýchto dvoch histogramov je možné použiť Battacharyyovu vzdialenosť, popísanú v sekcii 4.4. Už len s toho dôvodu, že súčet váh cez okno W_S nám dáva dobrý odhad skalárneho súčinu vektorov daných histognamov. Pravdepodobnostným modelom ohodnotenia vzoriek je teda rovnaký model ako v Y.Cai,2006 [8]:

$$p(y_t|x_t) = e^{-\frac{1}{2}\left(\frac{d(hist(x_t),hist^*(x_t))}{\lambda}\right)^2}$$

kde $hist(x_t)$ je histogram oblasti, ktorú určil MeanShift za daný segment. $hist^*(x_t)$ je referenčný histogram daného segmentu v mape. A $d(hist(x_t), hist^*(x_t))$ je ich Battacharyyovu vzdialenosť.

Váhu vzorky x_t konkrétneho segmetu je tak možné vyjadriť ako

$$w_t^i = p(y_t|x_t) * w_{t-1}^i$$

Váhy je nakoniec dobré znormovať tak, aby ich súčet cez vzorky každého segmentu bol rovný jednej. Takáto korekcia tak nakoniec uzatvára krok algoritmu. A model rozloženia pozícií vzoriek pri konkrétnom segmente by mal mať tvar

$$p(x_t|y_{0:t}) = \frac{p(y_t|x_t)p(x_t|y_{0:t-1})}{\int p(y_t|x_t)p(x_t|y_{0:t-1})dx_t}$$

Ak sa takýto postup aplikuje v jednom kroku pre všetky vzorky všetkých segmentov, tak dostávame ich novo ohodnotené a predĺžené trajektórie pohybu v relatívnom priestore robota. Do výpočtu pozície robota je pritom vhodné zahrnúť všetky vzorky spárované s ich príslušným segmentom spolu s ich váhami i napriek tomu, že jeden segment obsahuje niekoľko vzoriek. Tento výpočet pozície robota však nie je potrebný pri každom kroku filtrácie.

Ak má model pozície jedného segmentu príliš veľký rozptyl na snímku z kamery, tak sa segment v lokalizácii stratil a je z lokalizácie vypustený.

Problémom zostáva situácia, keď sa v obraze začínajú objavovať nové segmenty. Tento prípad, keď sa potrebuje nazrieť do mapy však taktiež nemusí byť vykonaný v každom kroku algoritmu a stačí, keď sa vykoná iba v prípade, že je segmentov málo na dobré určenie pozície robota.

Pri implementácii je vhodné obrázok po spätnej projekcii histogramu predspracovať konvolúciami tak, aby pri jednotlivých vzorkách daného segmentu nebolo treba tieto zdĺhavejšie operácie počítať a stačilo by nazrieť iba do predspracovaných polí. Veľkosť okna W_L a W_S pre MeanShift je

vhodné upravovať v jednotlivých krokoch tabuľkovo, v závislosti od vzdialenosti segmentu od robota. Aby sme zbytočne nezaťažovali procesor so vzorkami veľmi malej váhy, tak je ich dobré priebežne nahradzovať vzorkami umiestnenými do pozícií vzoriek s väčšími váhami. Tento krok sa nazýva prevzorkovaním. Priberanie nových segmentov je vhodné zahájiť napríklad iba v prípade, že zostal vzorkami sledovaný iba vopred zadaný malý počet segmentov. Tento počet podobne ako parameter λ sú charakteristické parametre scény. Parameter λ je použitý ako rozptyl pri zrovnaní histogramov, popisuje, ako moc sú si histogramy blízkych segmentov v scéne podobné. Pre veľké segmenty je dobré spočítať korekciu ich ťažísk, viď sekciu 4.5. Pre malé segmenty je táto korekcia zbytočná.

7.4 Navigácia k jednému bodu

V reálnom živote však málokedy potrebujeme vedieť svoju globálnu pozíciu. Stačí keď vieme, že sa nachádzame v okolí nejakého bodu, ktorého pozíciu v mape poznáme. V predchádzajúcich prípadoch sme vždy potrebovali aspoň dva také body, ktoré bolo možné vhodne projektovať do referenčnej roviny. Naskytá sa otázka, koľko informácií stratíme, ak máme iba jeden takýto bod. Odpoveď je jednoduchá pozícia robota bude určená vzdialenosťou a natočením vzhľadom k tomuto bodu. S takto zjednodušenou pozíciou je možné pracovať takmer rovnako ako s globálnou pozíciou. S tým rozdielom, že neoznačuje iba jeden bod v mape, ale celú kružnicu možných polôh. Každý bod na tejto kružnici má určené pevné globálne natočenie, ktoré sa získa pripočitaním relatívneho natočenia (vzhľadom k navigačnému bodu) k natočeniu priamky prechádzajúcej danou pozíciou a navigačným bodom.

Vzdialenosť k bodu je možné vyjadriť ako $h \cdot \|(r, s)\|$, kde (r, s) sú jeho súradnice v referenčnej rovine a h je jeho hĺbka, tj. rozdiel výšky kamery od výšky bodu v scéne. Natočenie k navigačnému bodu je natočením bodu v referenčnej rovine od osi y. Teoreticky tak stačí sledovať na sekvencii snímkov z kamery iba daný segment obsahujúci navigačný bod. Preto je vhodné takýto bod voliť ako ťažisko dobre identifikovateľného malého segmentu. Dobrou identifikáciou sa myslí, že jeho okolie neobsahuje podobné farby a algoritmus Mean-Shift ho tak môže dobre sledovať medzi jednotlivými snímkami.

Kapitola 8

Záver

Výsledkom práce je analýza a návrh všeobecných postupov pri riešení daného problému. Implementácia sa preto môže pri rôznych mapách líšiť. Napríklad lepšou identifikáciou segmentov, ktorá v práci prebieha pomocou histogramov.

Okrem vyhľadania a spojenia vhodných postupov riešenia častí problému som prácu obohatil hlavne o vyjadrenia parametrov pozície kamery zo vzťahov perspekívnej projekcie (kapitola 2). Prevodom a počítaním pozície robota pomocou referenčnej roviny z identifikovaných bodov obrazu s 3D bodmi v mape (sekcia 3.1). Korekciou ťažísk segmentov(sekcia 4.5). Párovaním identických bodov(kapitola 5). Výberom reprezentatívnej štruktúru mapy (kapitola 6). Vyjadrením váh posunutí segmentov z algoritmu MeanShift v sekcii 7.1. A upravením algoritmu Monte Carlo filtrácie viacerých segmentov zverenjeným v zborníkoch Computer Vision - ECCV 2006 [8].

Aplikácií popísaného postupu lokalizácie kamery je niekoľko. Prostredie môže byť pozmenené pomocou dobre farebne rozlíšiteľných značiek, tak aby na jednom snímku sa vyskytovalo čo najmenej segmentov farby nejakej značky. To je z dôvodu časovej zložitosti pri hľadaní počiatočnej polohy robota v mape, viď algoritmus 3. Takéto značky si možno predstaviť ako malé segmenty špecifickej farby. Malé preto, aby nebolo treba väčšiu korekciu ich ťažísk, viď sekcia 4.5. A tiež preto, aby boli dobre uchopiteľné algoritmom MeanShift (sekcia 7.1) pri pohybe robota. Ich farba by sa nemala príliš vyskytovať v ich okolí na obrázku z kamery. A samotné sledovanie pozícii už ponechať na sledovaní segmentov a výpočte pozícií pri spárovaní ich súradníc s odpovedajúcimi súradnicami v mape (sekcia 3.1).

V takomto prostredí si možno napríklad predstaviť plne automatický

vysavač, ktorý sa lokalizuje na základe značiek na strope, resp. na stenách Alebo autonómnu kosačku, ktorá sa lokalizuje podľa značiek na vzdialených stromoch, rep. budovách.

Literatúra

[1] Daniel F. DeMenthon and Larry S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, vol. 15, no. 1 pp. 123-141, Máj 1995. [cited at p. 16]

[2] Olivier Faugeras. *Three-Dimensional Computer Vision, A Geometric Viewpoint.* MIT Press, 1993. [cited at p. 46]

[3] Andrew W. Fitzgibbon. Robust registration of 2d and 3d point sets. *In The British Machine Vision Conference*, 2001. [cited at p. 45]

[4] Intel Corporation. *Open Source Computer Vision Library.* [cited at p. 3, 4, 17, 20]

[5] J.R.Parker. *Algorithms For Image Processing And Computer Vision.* Whiley, 1997. [cited at p. 20]

[6] Long Quan and Zhongdan Lan. Linear n-point camera pose determination. *Trans. on Pattern Analysis and Machine Intelligence*, VOL. 21, NO. 7, Júl 1999. [cited at p. 16, 17, 18]

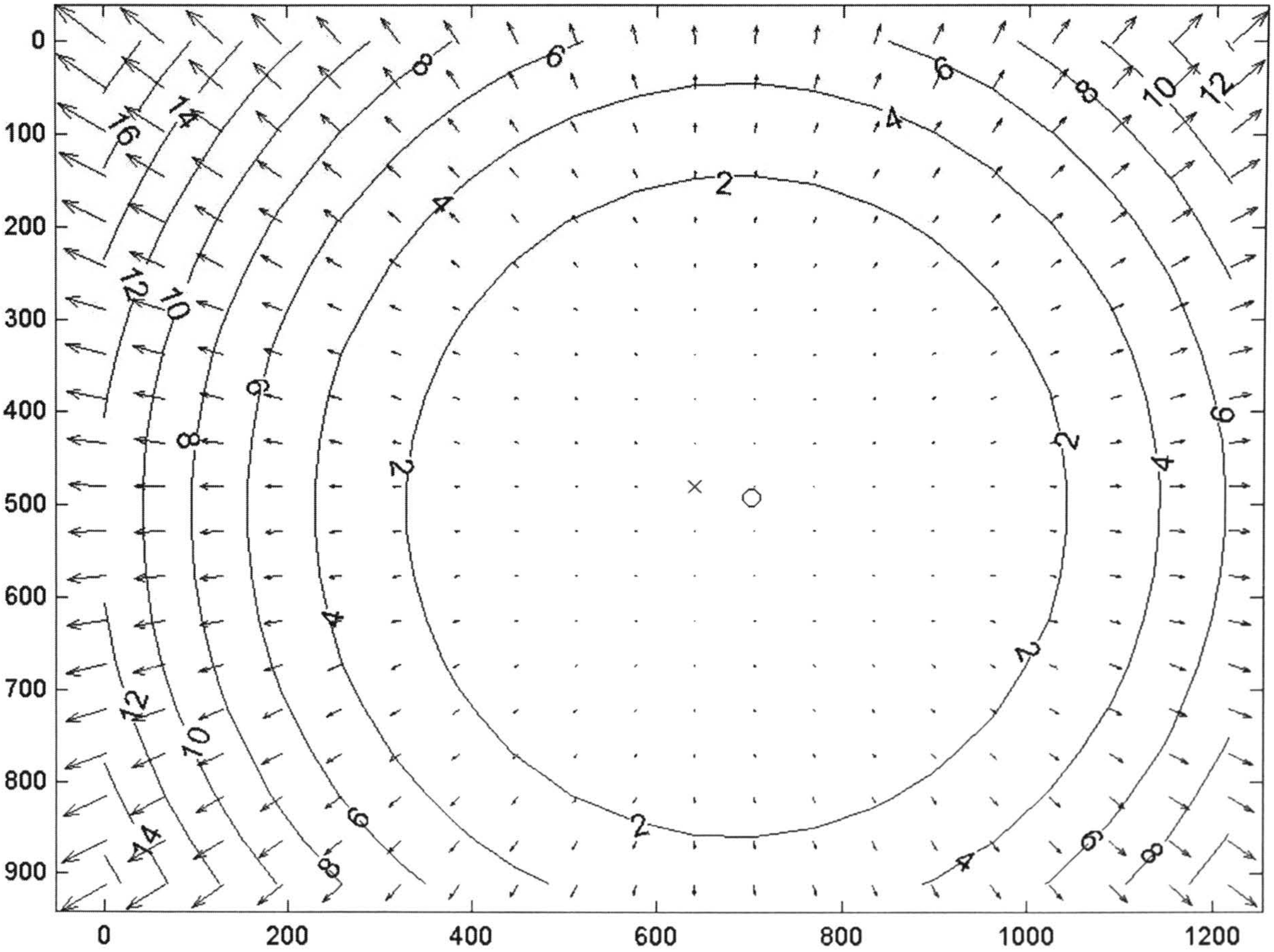
[7] T. Kanade S.Baker, A.Datta. Parametriziting homographies. Január 2006. [cited at p. 7]

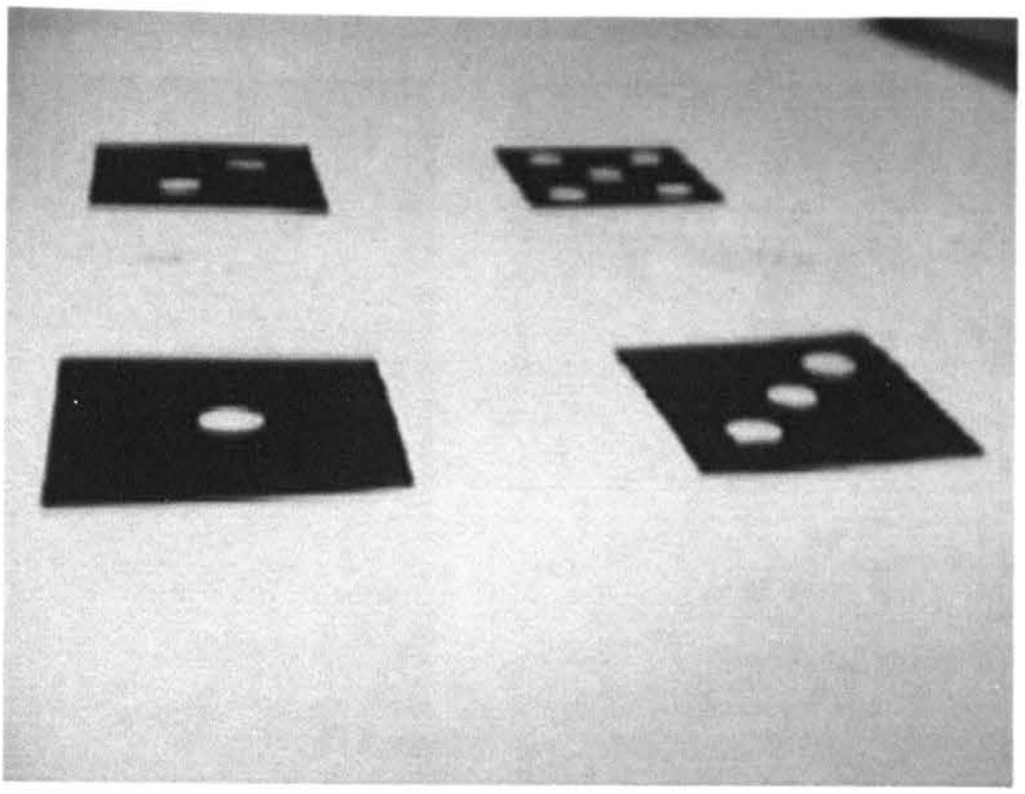
[8] James J. Little Yizheng Cai, Nando de Freitas. Robust visual tracking for multiple targets. *Computer Vision - ECCV 2006*, Máj 2006. [cited at p. 43, 46, 48, 50]

Zoznam obrázkov

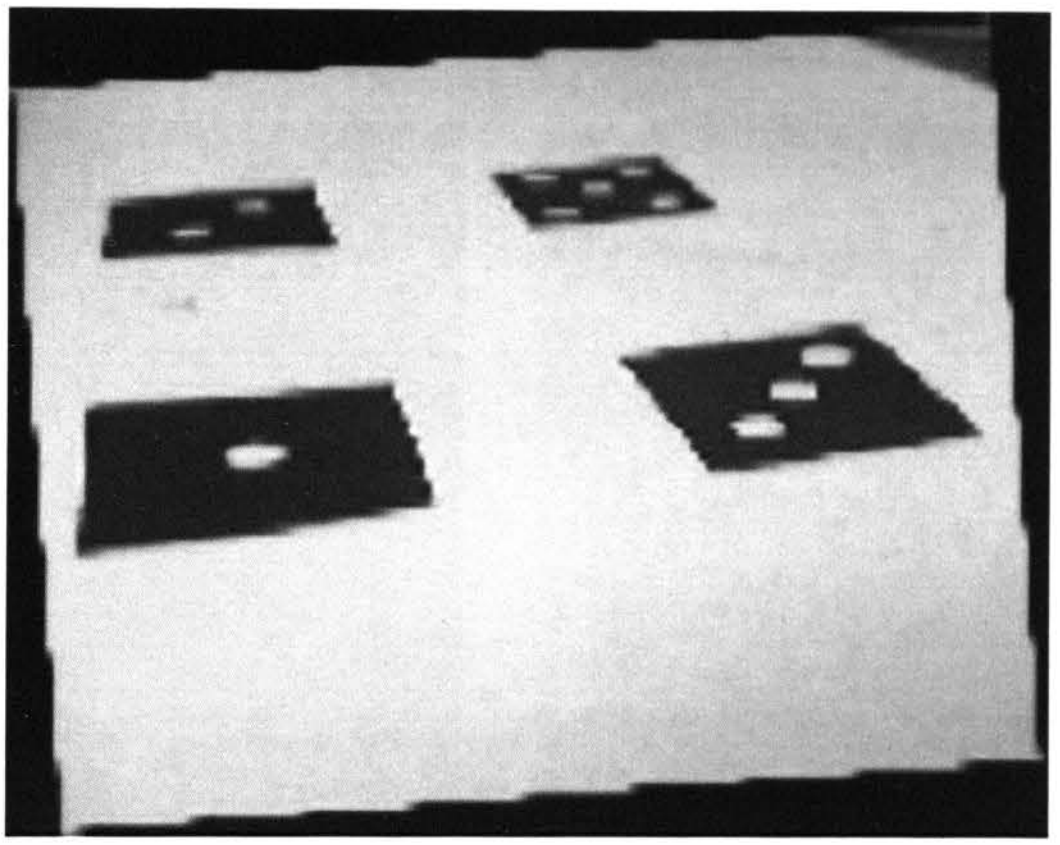
Prílohy

Skreslenie Kamery

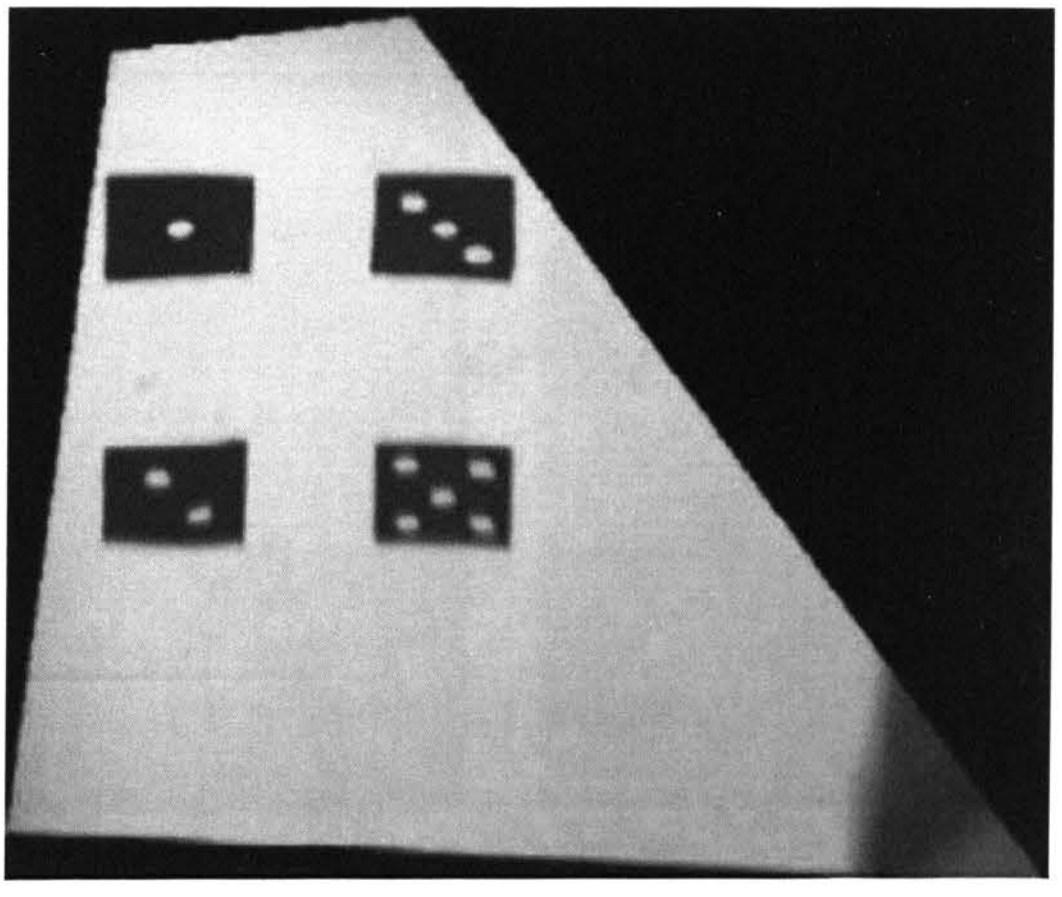




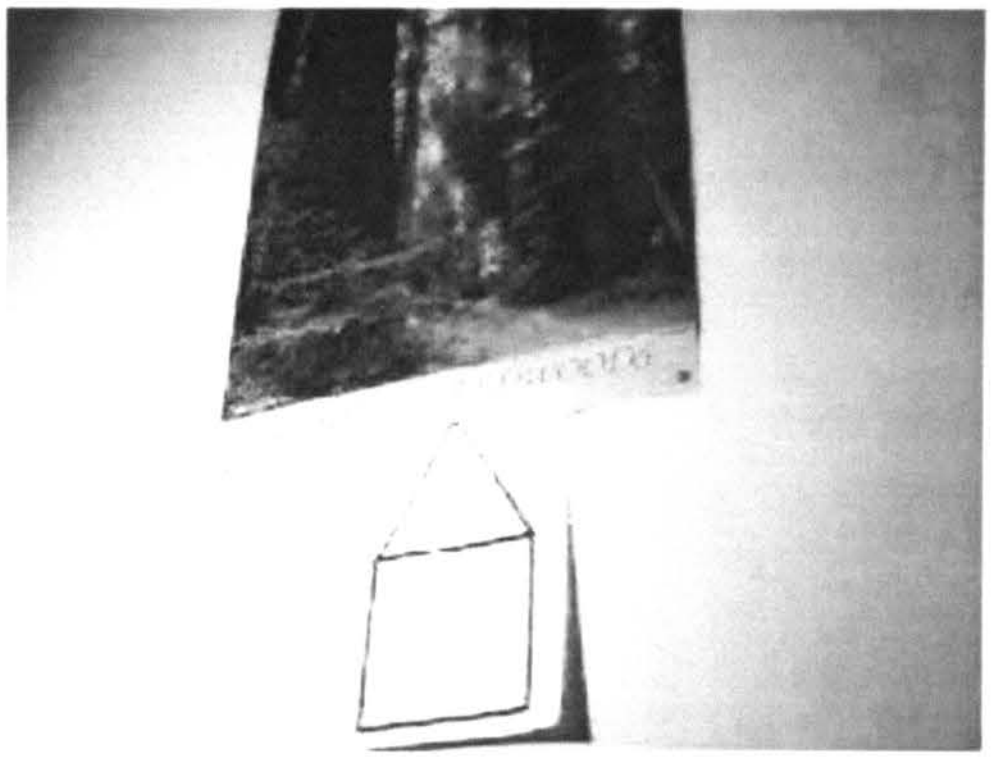
Pohľad z kamery do scény

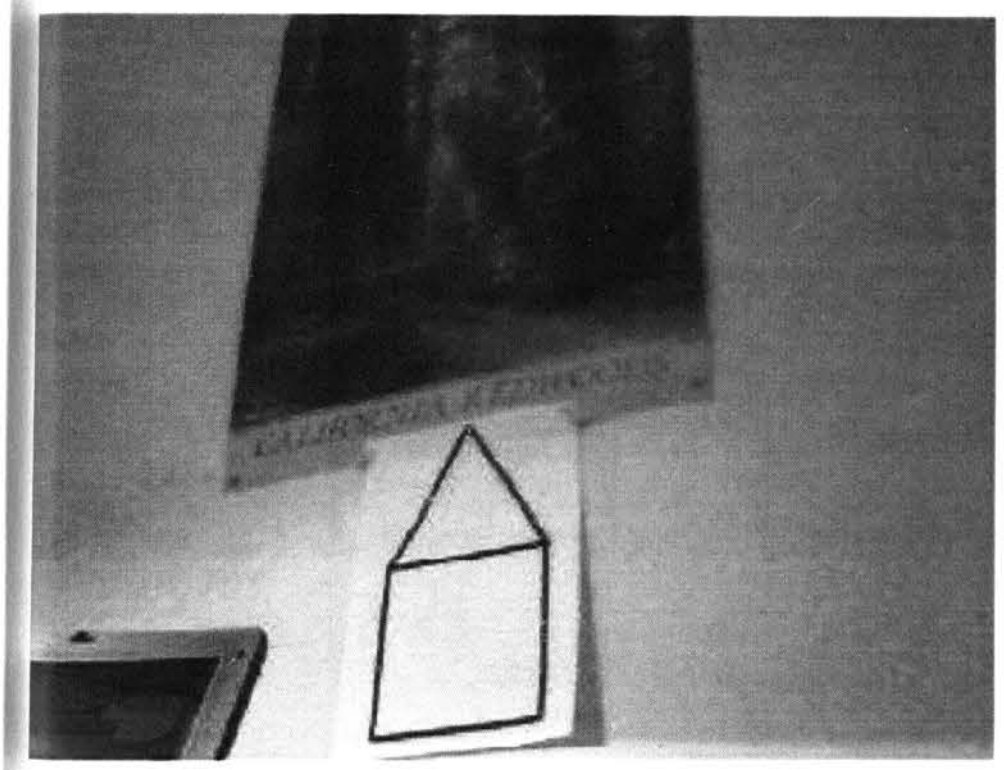


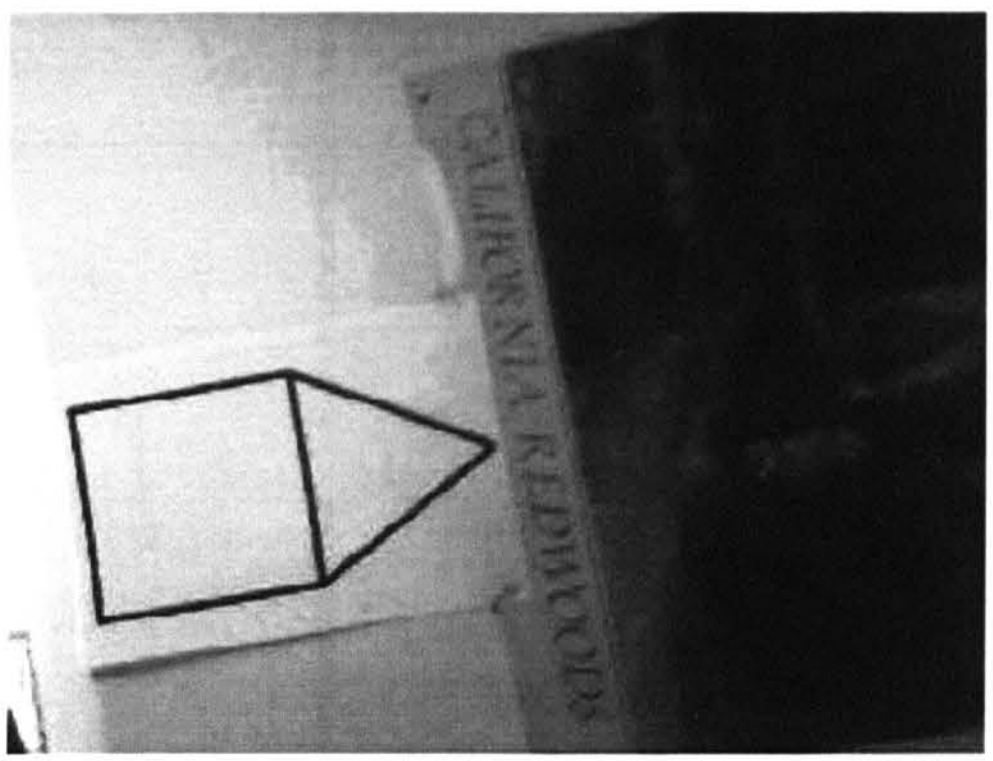
Korekcia pohľadu natočením okolo osi pohľadu

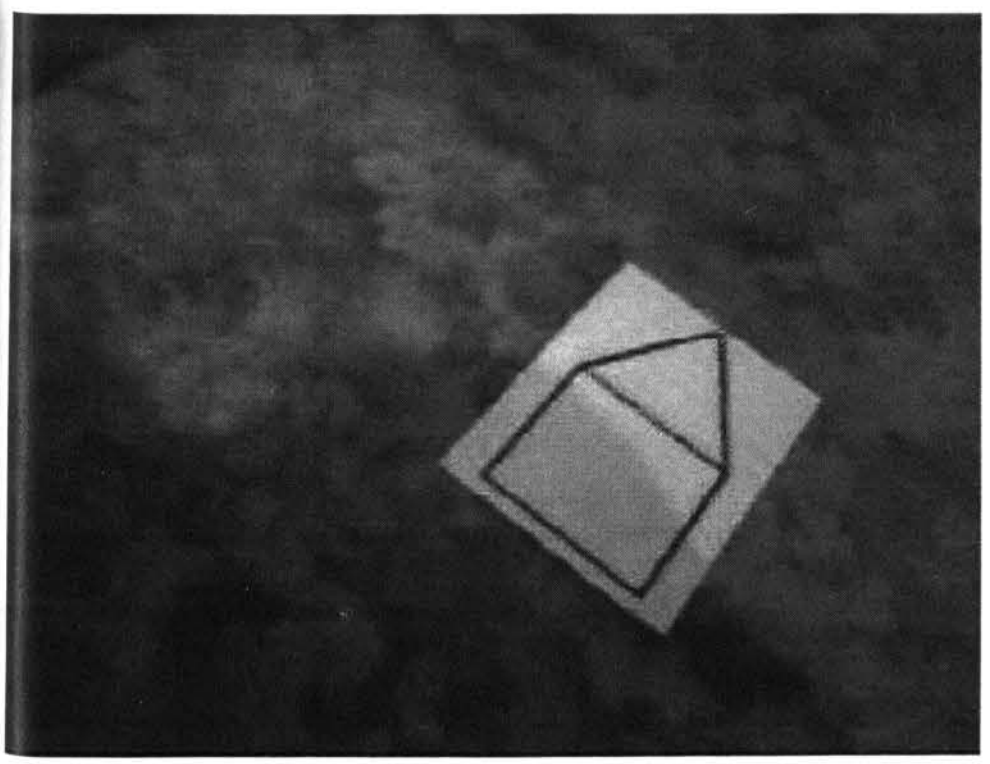


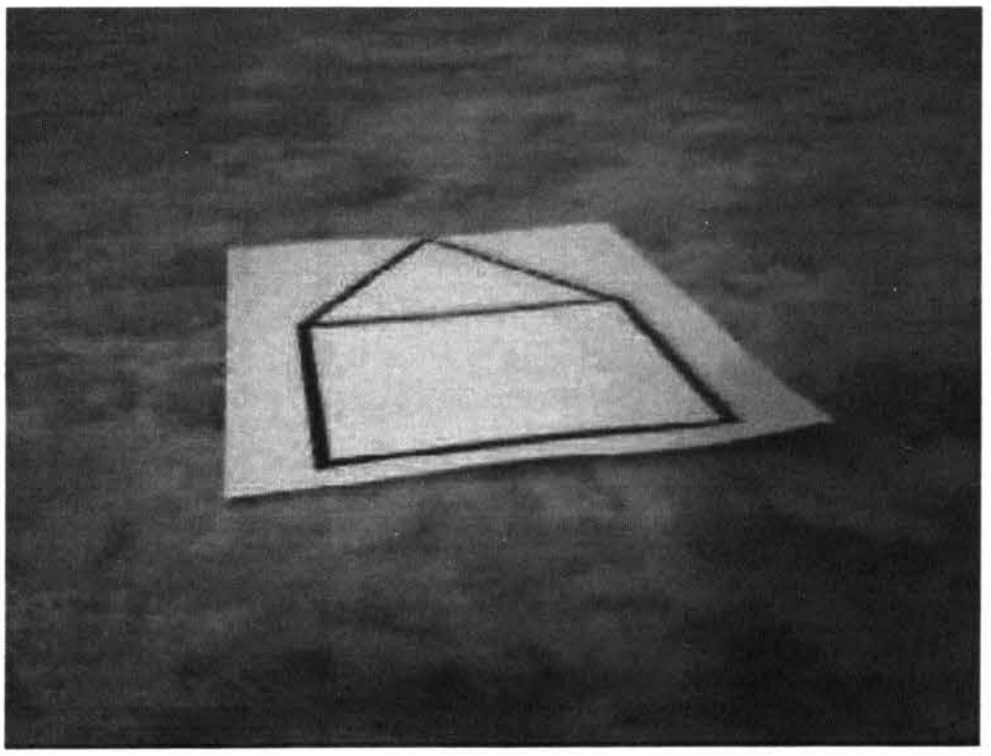
Zobrazenie snímku v referenčnej rovine



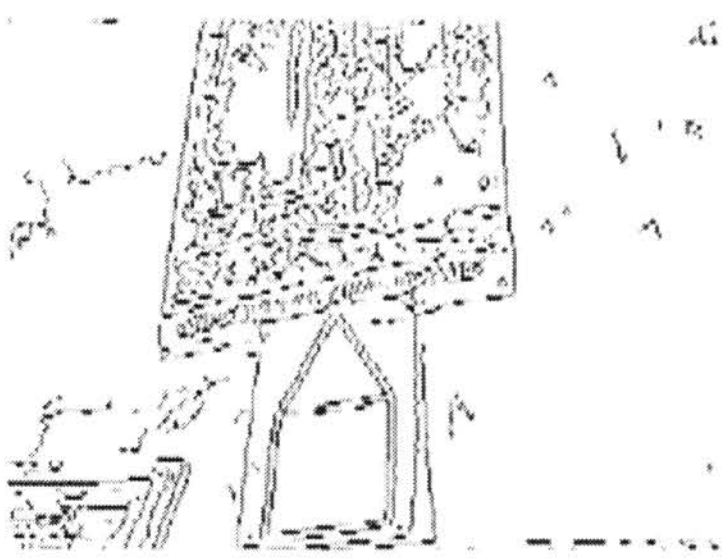


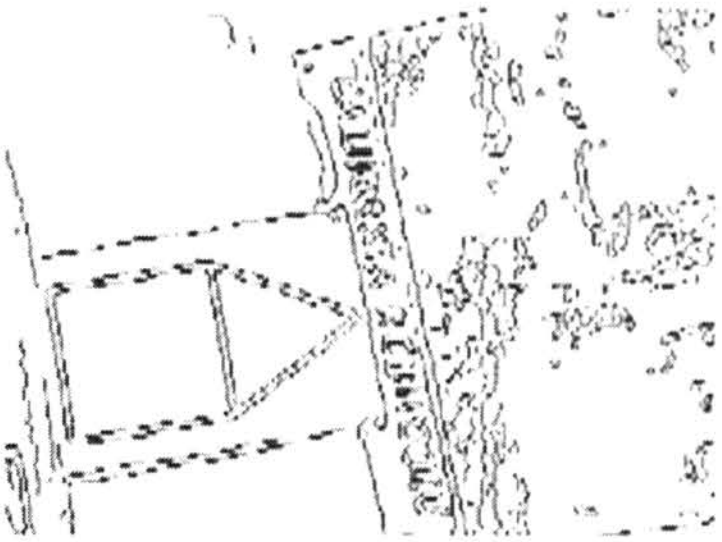


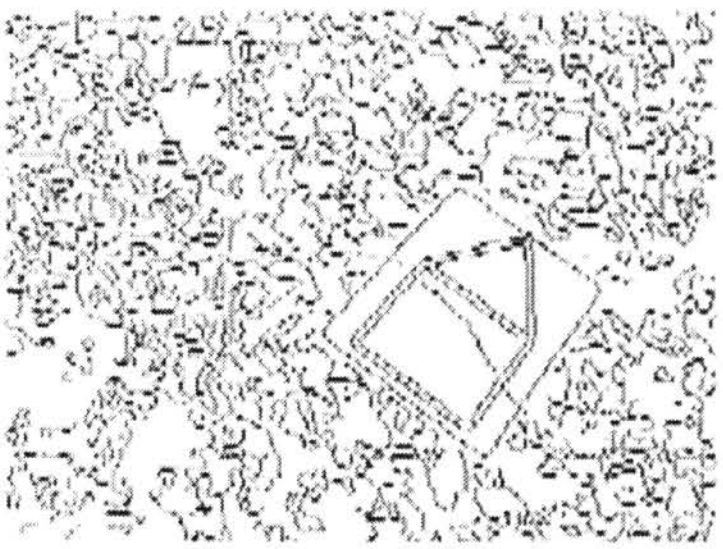


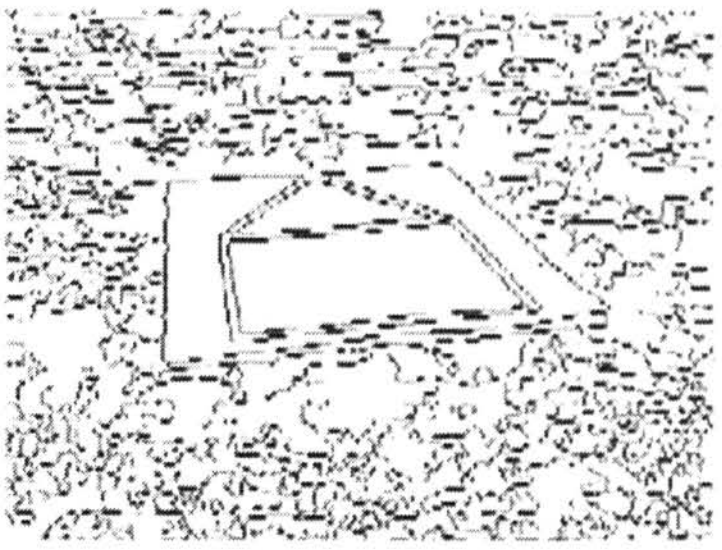


Pôvodné obrázky použité na spracovanie obrazu

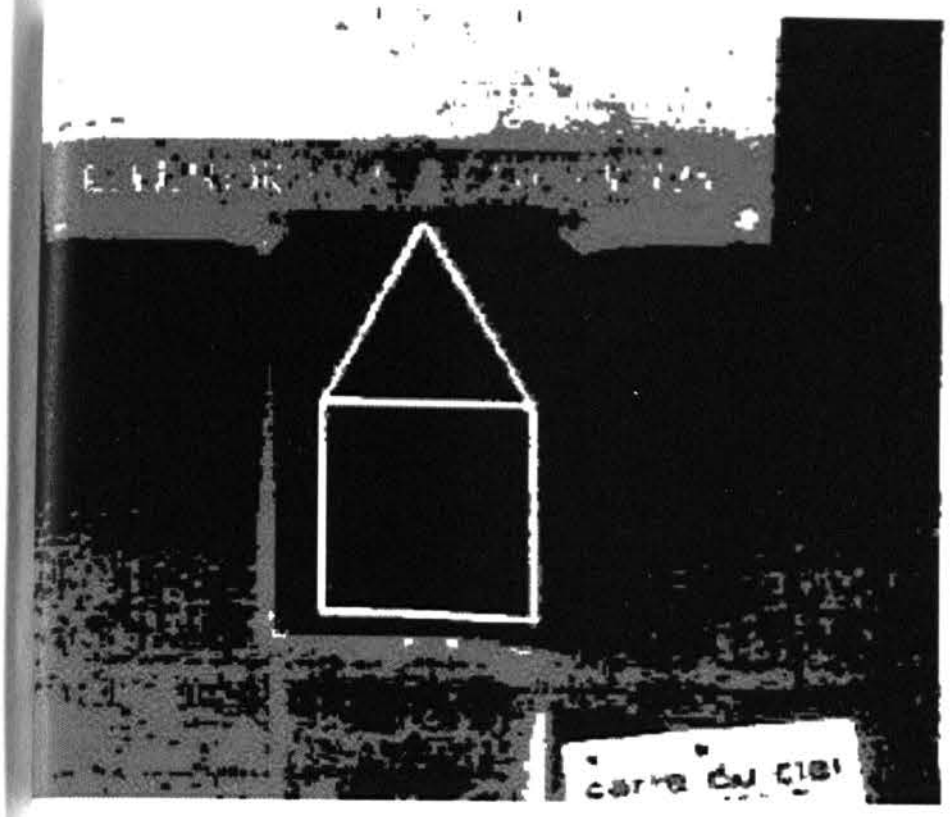


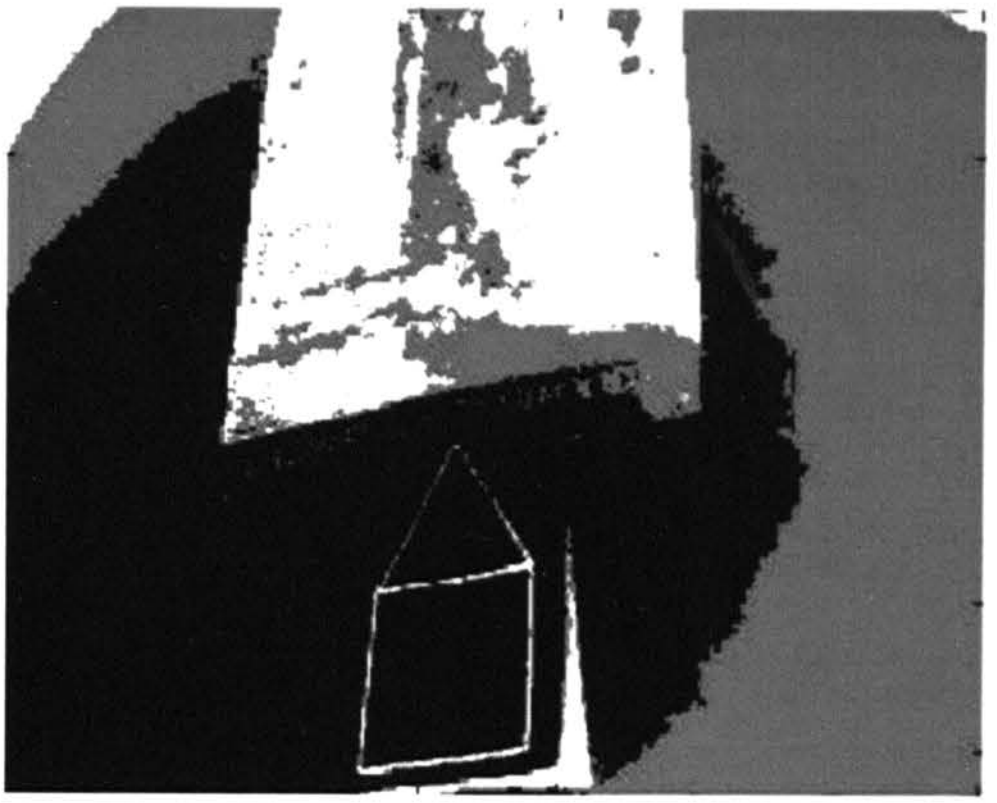


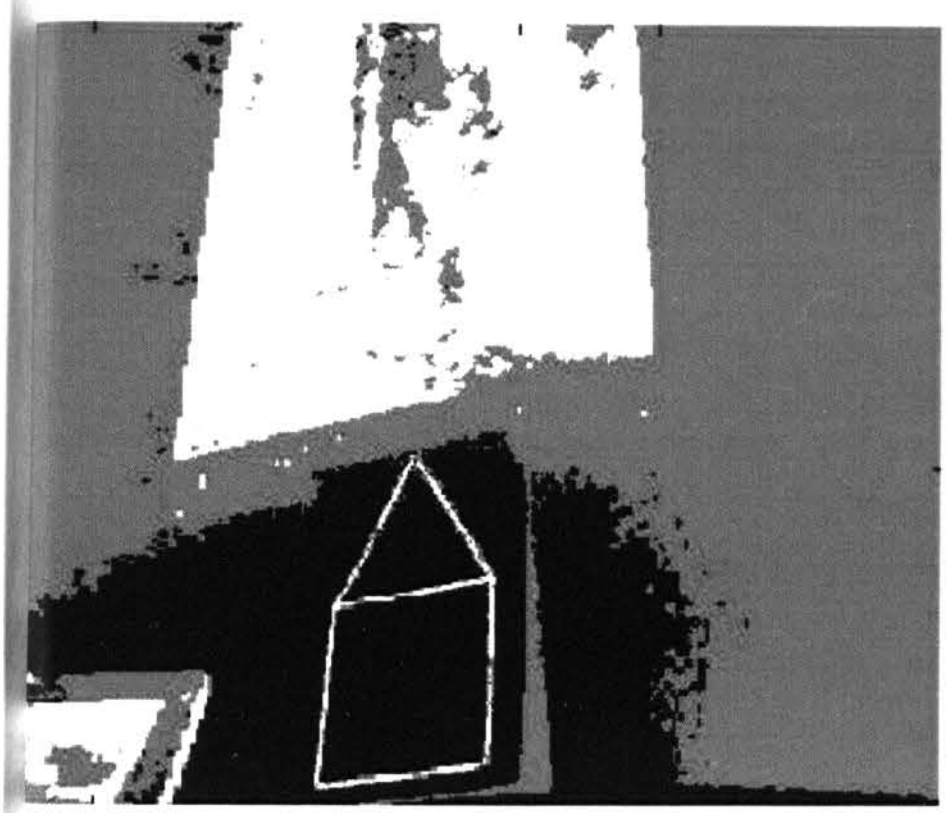


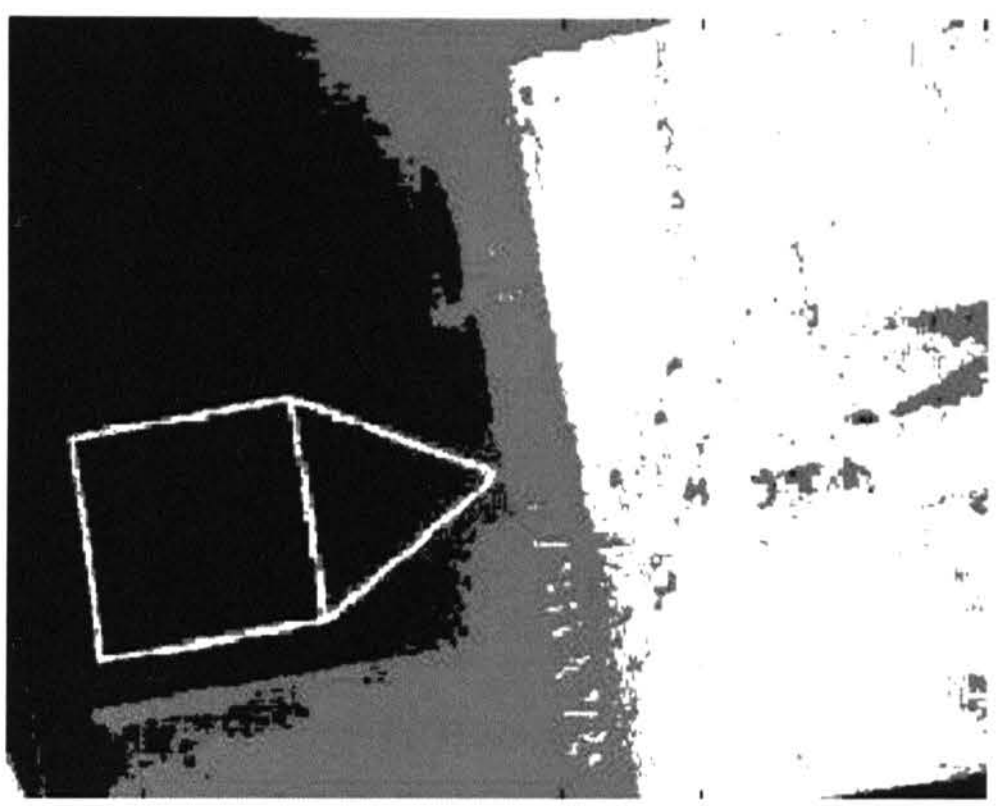


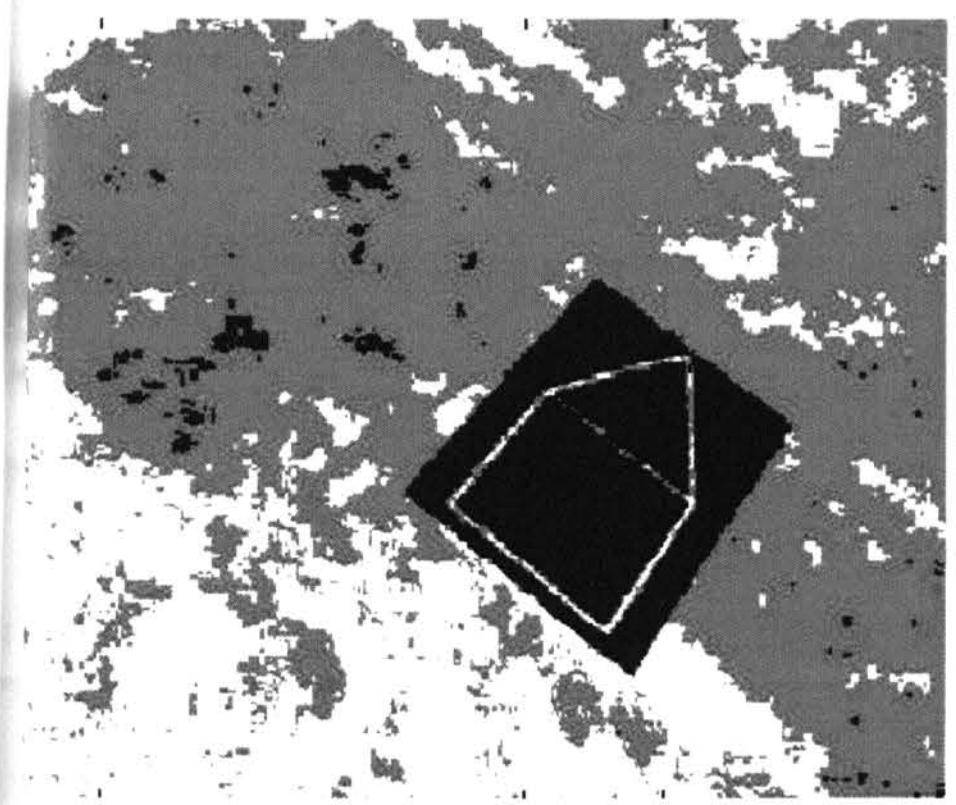
Detekcia hrán pomocou Cannyho hranového detektoru

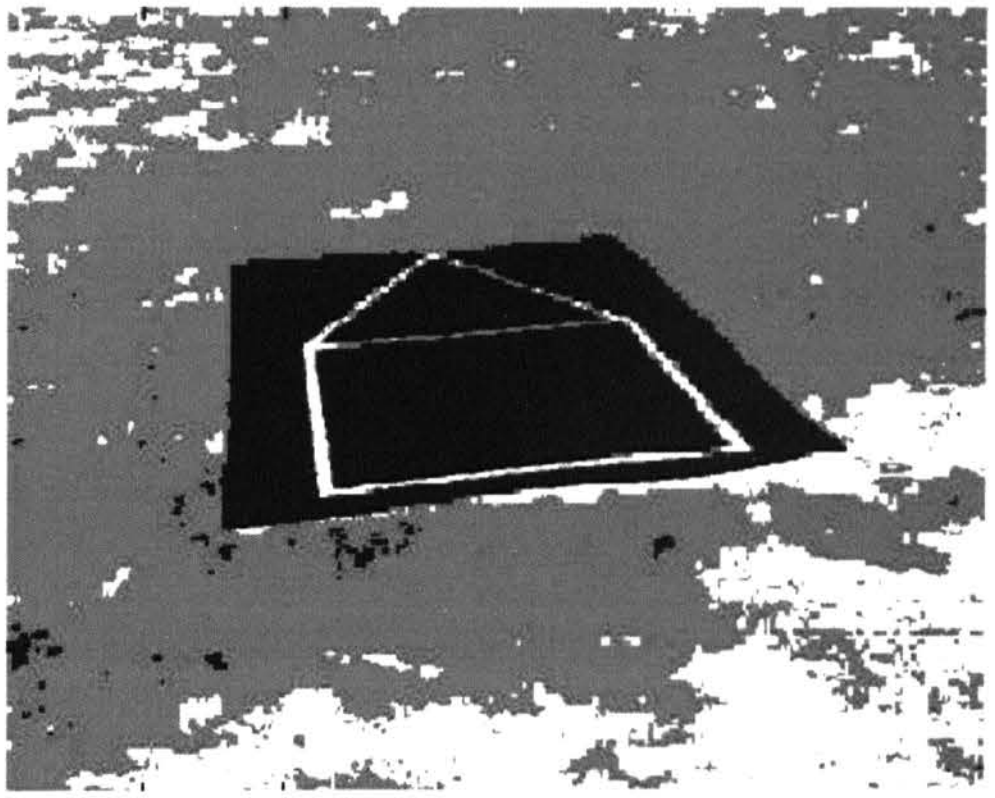




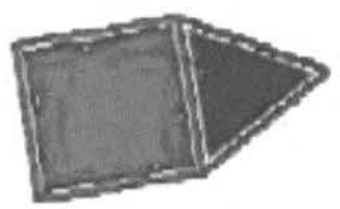




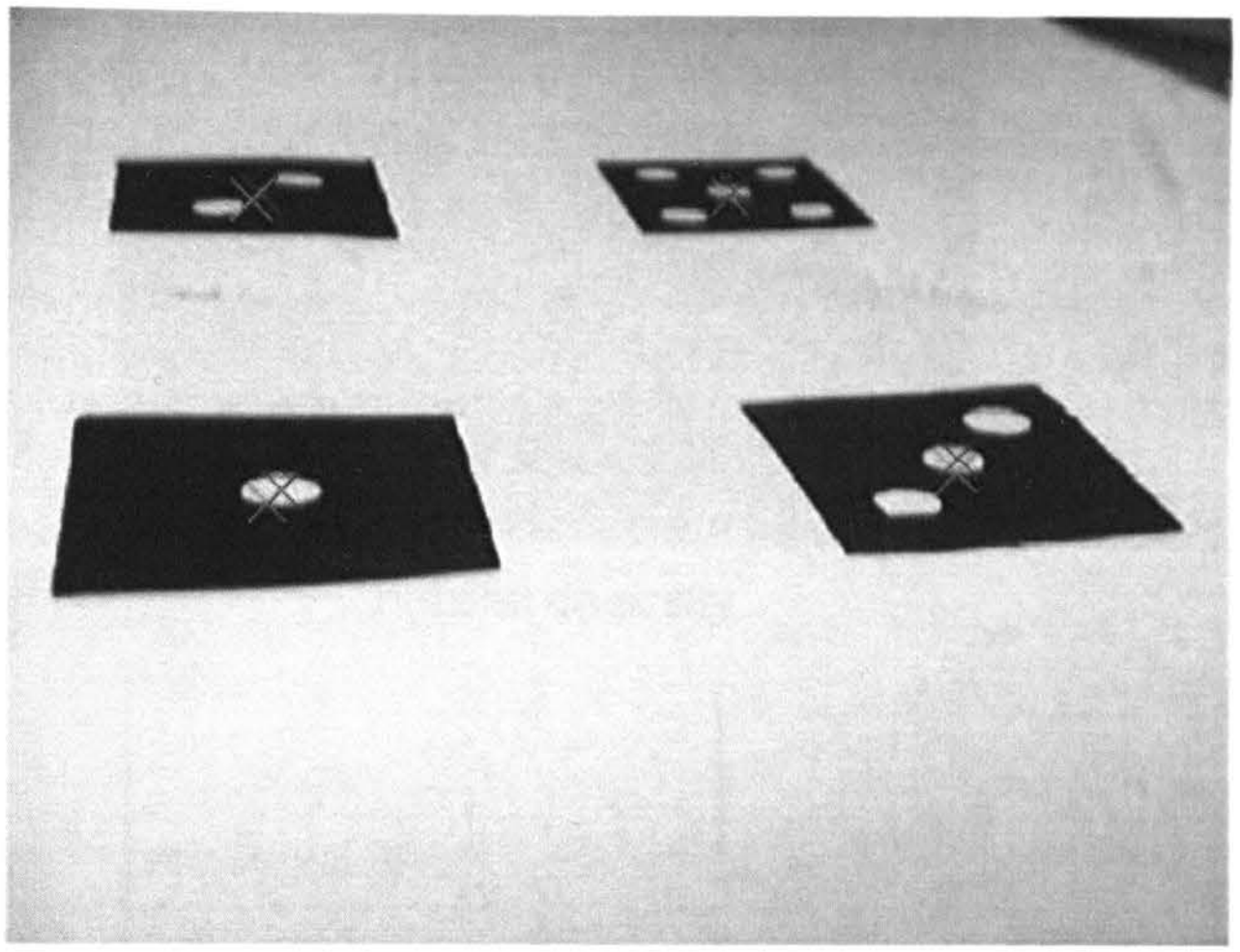




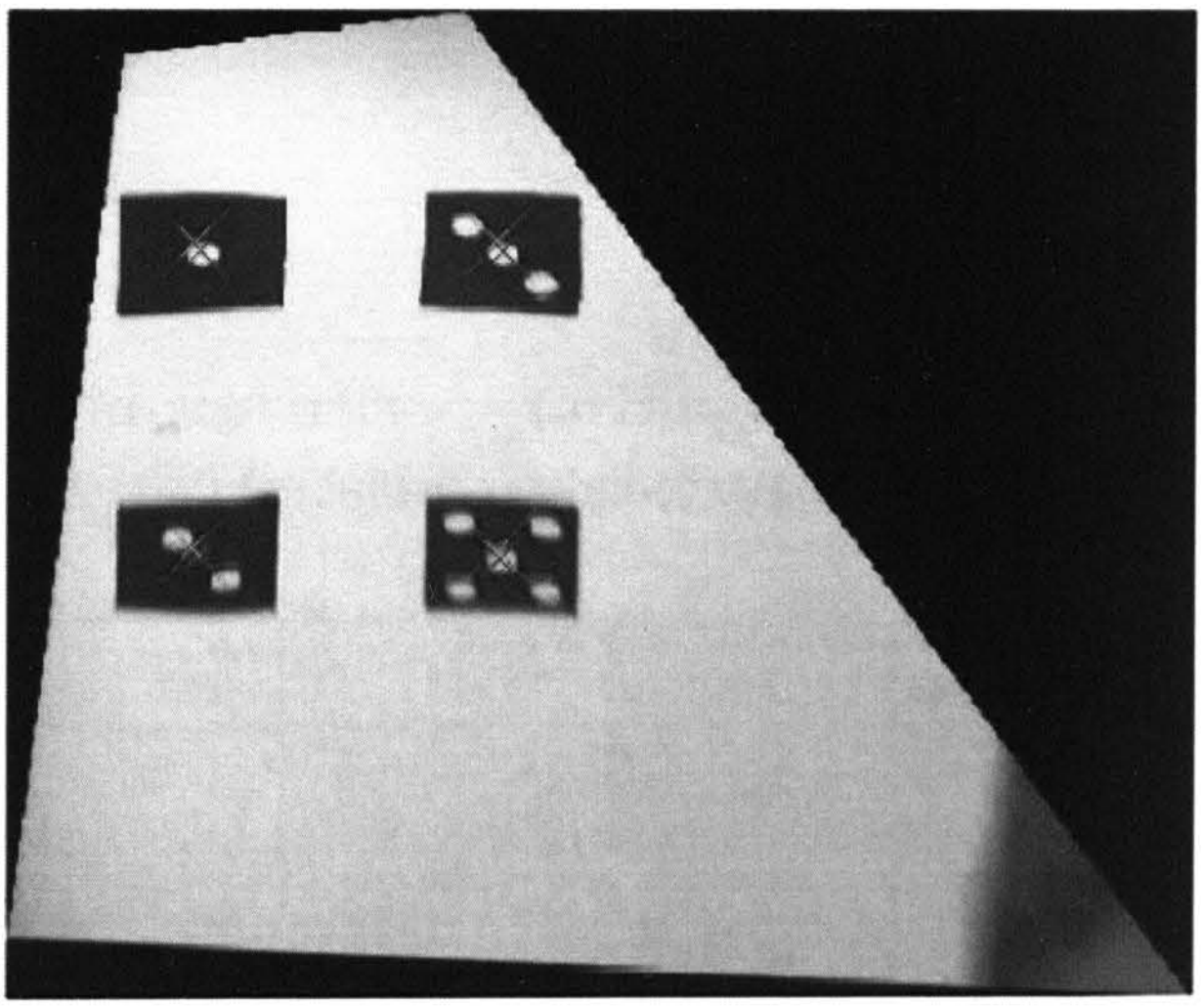
Prahová segmentácia



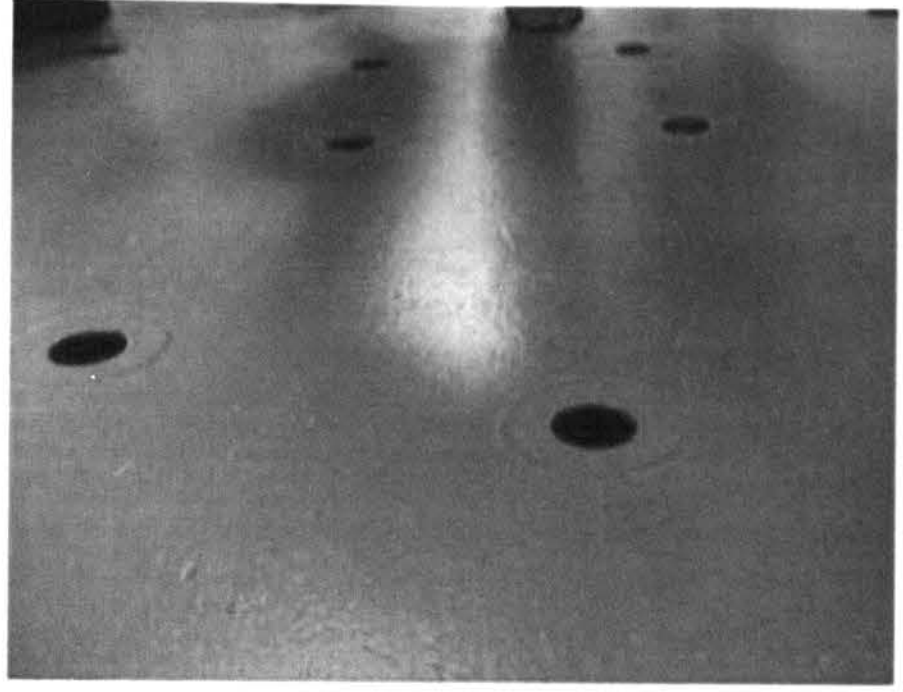
Hranová segmentácia



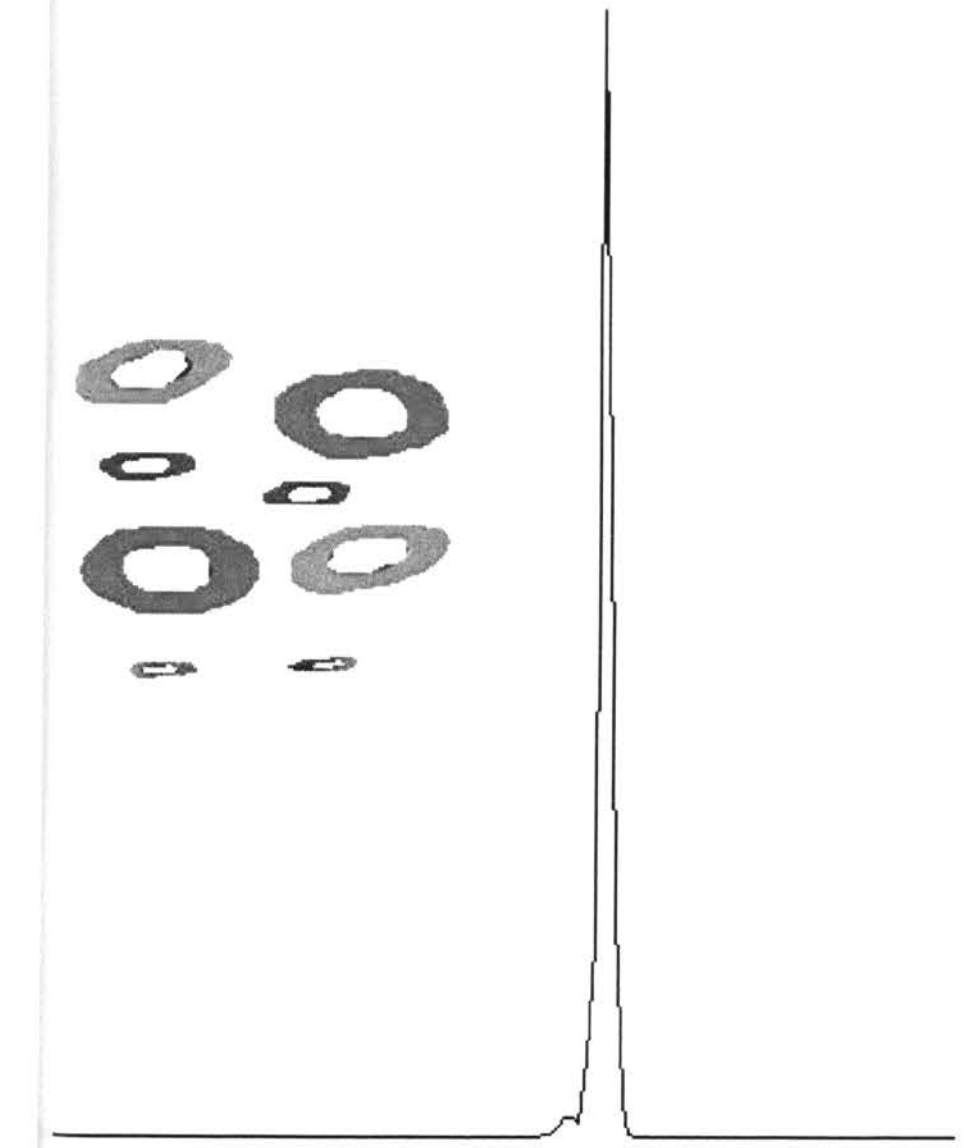
Ťažiská segmentov na snímku ako aritnetické priemery súradníc ich bodov sú vyznačené zeleno, zatiaľ čo projektované ťažiská z mapy sú vyznačené červeno.



Situácia v kolmom pohľade na rovinu segmentov



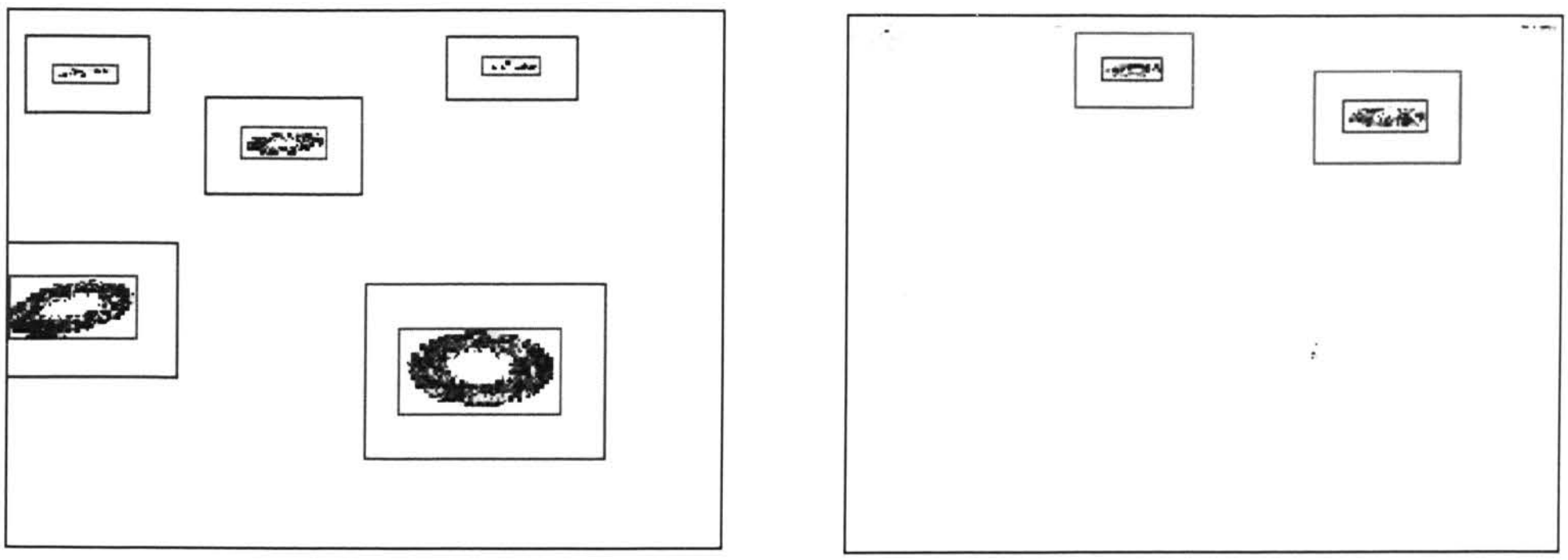
Pohľad do scény



stogram červených vzoriek segmentov

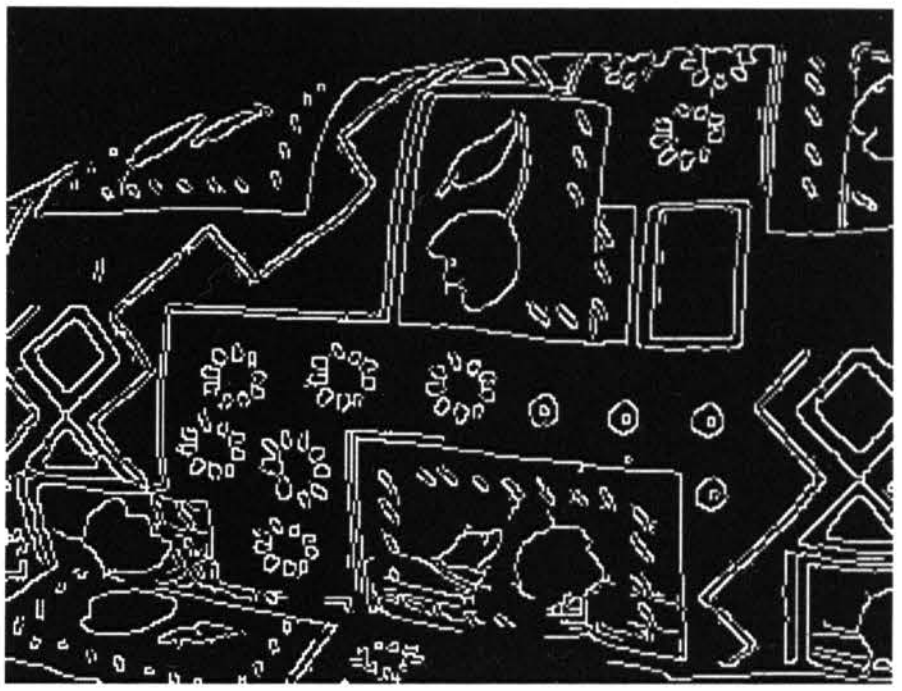
Histogram modrých vzoriek segmentov

(zo zložky farby určujúcu farebnosť v modeli HSV)

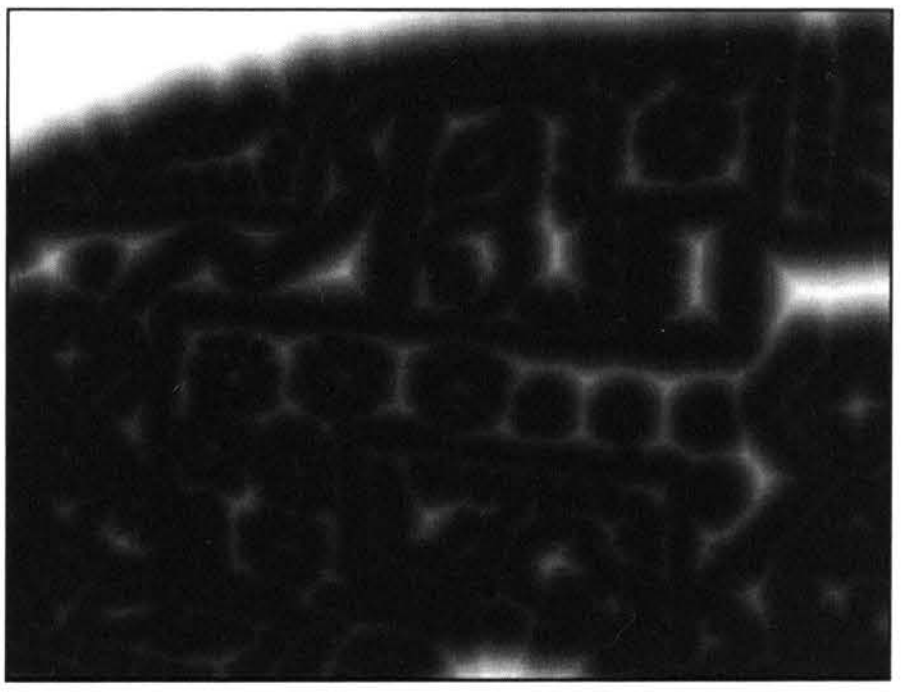


Spätná projekcia histogramov a
obálky pre algoritmus MeanShift

Pohľad z kamery



Detekcia hrán



Odkazy na najbližšie hrany pripravené na zrovnávanie pohľadov do mapy algoritmom ICP