ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

fakulta Elektrotechnická
katedra Kybernetiky

Kalibrace lokalizačního vizuálního systému

Calibration of localization visual system

Bakalářská práce

Studijní program: Kybernetika a robotika
Studijní obor: Robotika

Vedoucí práce: Ing. Jan Chudoba

Václav Moravčík **Praha 2014**

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Václav M o r a v č í k

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Kalibrace lokalizačního vizuálního systému

Pokyny pro vypracování:

1. Seznamte se s lokalizačním systémem robotické platformy SyRoTek, využívající zpracování obrazu z kamery umístěné nad pracovní plochou, po které se pohybují lokalizované roboty.
2. Navrhněte metodu kalibrace transformace mezi obrazem z kamery a reálnými souřadnicemi v pracovní ploše. Pro určení vzájemné polohy mezi kamerou a pracovní plochou využijte detekci hranice plochy v obraze, nebo využijte uměle instalované dobře detekovatelné značky, které poslouží k získání transformace.
3. Navrženou metodu implementujte a začleňte do stávajícího lokalizačního systému.

Seznam odborné literatury:

[1] Hlaváč V., Sedláček M.: Zpracování signálů a obrazů, skriptum, Vydavatelství ČVUT 2009.
[2] Hlaváč V., Šonka M.: Počítačové vidění, Grada, Praha, 1992.

Vedoucí bakalářské práce: Ing. Jan Chudoba

Platnost zadání: do konce letního semestru 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 1. 2014

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne ……. …….

Podpis autora práce

Poděkování

Na tomto místě bych chtěl poděkovat Ing. Janu Chudobovi za odborné vedení práce, cenné rady a připomínky, čas strávený konzultacemi a především také za trpělivost, kterou se mnou měl. Dále bych chtěl poděkovat rodině a přátelům za jejich podporu, při psaní této práce.

Abstrakt

Cílem této práce je navrhnout metodu kalibrace transformace visuálního lokalizačního systému, který je součástí většího systému pro výuku mobilní robotiky SyRoTek. V první části práce je popsána teorie digitalizace obrazu, zpracování digitálního obrazu a projektivní transformace obrazu. Největší prostor první části je věnován teorii detekce hran v obraze konvolučními operátory. Jádro práce tvoří návrh a zpracování metody získání transformace mezi obrazem kamery a hřištěm systému SyRoTek. Tato část obsahuje popis všech hlavních metod a algoritmů použitých při implementaci programu. Na závěr jsou vyhodnoceny experimenty testující funkčnost a přesnost vytvořené metody v reálných podmínkách.

Klíčová slova: digitální obraz, zpracování obrazu, detekce hran, konvoluční operátory, 2D homografie

Abstract

The goal of this bachelor thesis is to implement a method of calibration a visual localization system which is part of the larger system for teaching mobile robotics called SyRoTek. In the first part of the work theory of image digitalization, image processing and projective transformation of image are discussed. Most space of the first part is dedicated to theory of edge detection with the use of convolution operators. The core of this work consists of design and implementation of method for obtaining a transformation between image from the camera and SyRoTek arena. This part consists of description of alt the main methods and algorithms used in implementation of the program. In conclusion the experiments testing the functionality and precision of created method in real conditions are evaluated.

Key words: digital image, image processing, edge detection, convolution operators, 2D homography

Obsah

1. Úvod

V dnešní době se stále více objevují snahy o zatraktivnění výuky. Katedra Kybernetiky FEL ČVUT je pro vytvoření atraktivních kurzů pro studenty vhodným místem. Výuka je tu často doplněna o praktickou část, která je pro studenty vždy lákavá, a zároveň si na ní studenti lépe osvojí teorii. Jedním z projektů, které kladou důraz na praktickou stránku výuky, je systém SyRoTek (System for Robotic e-learning). Jedná se o systém pro praktickou výuku robotiky, obsluhovatelný přes webové rozhraní. Systém se skládá z robotické arény, ve které se roboty pohybují, a z několika serverů zajišťujících chod systému. Systém funguje tak, že studenti nahrají jimi vytvořený program na server, roboty poté vykonají tento program a systém následovně vyhodnotí, zda bylo splněno zadání úlohy. K tomu, aby mohl být pohyb robotu vyhodnocen, slouží lokalizační podsystém, který kamerou umístěnou nad hřištěm snímá pohyb robotů. Pro vyhodnocení pohybu robotu je také důležité získávání transformace mezi obrazem a plochou hřiště. V současné chvíli je tato transformace staticky určena. Pro potřeby systému by ale bylo vhodnější, aby byl schopen lokalizační podsystém určit tuto transformaci dynamicky s využitím obrazu z kamery. Cílem mé práce je navrhnout metodu, která by umožnovala dynamické získávání transformace resp. kalibraci této transformace.

První část práce (kapitoly 2 – 4) se soustředí na popsání a vysvětlení teorie, na které je postavena praktická část. Pro pochopení teorie první části práce je předpokládaná určitá úroveň znalostí, především z oblasti matematiky. V druhé kapitole je stručně popsán vznik digitálního obrazu a jeho důležité vlastnosti. Třetí kapitola je věnována hranám v digitálním obraze, a především jejich detekci v obraze konvolučními operátory. V této kapitole jsou také podrobněji rozebrány některé hranové detektory, pracující s konvolučními operátory. Na konci této kapitoly jsou pak naznačeny některé metody pro další zpracování detekovaných hran. Poslední teoretická kapitola (kapitola 4) stručně uvádí do problematiky projektivní geometrie a projektivní transformace ve 2D.

Druhá část práce je zaměřena na návrh a implementaci programu, který kalibruje transformaci mezi obrazem z kamery a plochou hřiště. V páté kapitole je postupně rozebrán krok po kroku postup, kterým jsou v obraze detekovány nejdříve hrany, ze kterých jsou dalším zpracováním získány významné body obrazu. Tyto významné body jsou následovně využity při výpočtu transformace. Tato kapitola také obsahuje popis různých jak běžně využívaných, tak i pouze pro tuto konkrétní úlohu využitelných

algoritmů, které byly pro implementaci programu využity. V šesté kapitole jsou pak prezentovány poznatky, které byly využity pro odladění programu. Dále naměřená data, která byla získána testováním programu v reálných podmínkách. Tyto data, resp. výsledky jsou na závěr diskutována.

2. Digitální obraz

Svět, ve kterém se pohybujeme a který pozorujeme, je spojitý. Pokud tedy chceme zaznamenat obraz, resp. vícerozměrný signál a dále ho zpracovávat pomocí počítače, je zapotřebí obraz digitalizovat. Při digitalizaci obrazu využíváme dvou samostatných procesů: *vzorkování* a *kvantování*. Při těchto procesech dochází k diskretizaci obrazové funkce $f(x, y)$ a s tím spojeným částečným ztrátám informace.

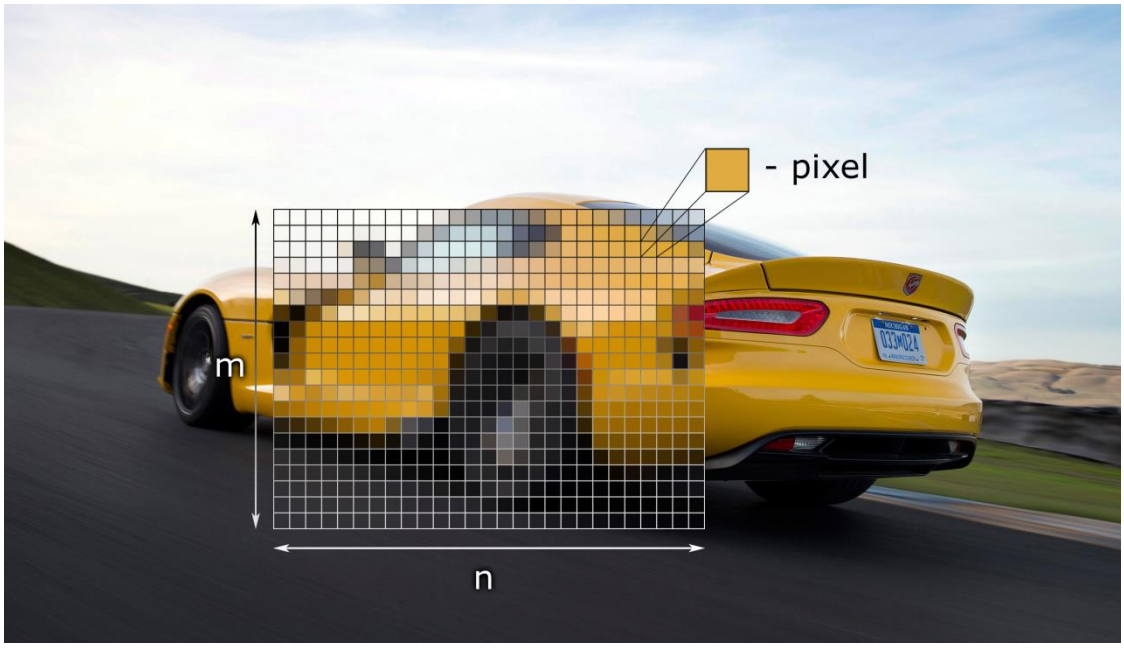
2.1. Vzorkování

Vzorkování je proces, při kterém dochází k transformaci ze spojitého časoprostoru do diskrétního časoprostoru. Obrazová funkce je rozdělena do uniformních, většinou čtvercových částí, které se nazývají pixely. Podle detailů, které májí být v obraze vidět, musíme upravit velikost pixelů. Tato skutečnost je popsána Shannonovým vzorkovacím teorémem, který říká, že vzorkovací frekvence musí být alespoň dvakrát větší než největší frekvence v obraze [1, s. 4-5, 114]. U digitálního obrazu, který je ve výsledku nejčastěji pozorován člověkem, je potřeba, aby obraz především „vypadal dobře“. To znamená, aby ztráta informace při digitalizaci nebyla příliš viditelná [2, s. 7-9]. Na obr. 1 je vidět, jak je plocha obrazu diskretizována na jednotlivé pixely v matici m x n.

2.2. Kvantování

U digitálního obrazu nás většinou zajímá intenzita signálu, respektive intenzita obrazové funkce. Zařízení pořizující digitální obraz může například měřit intenzitu celého barevného spektra, potom se jedná o obraz v tzv. stupních šedi (v angličtině „grayscale“). Nejčastěji se praxi ale setkáváme s barevným obrazem, u kterého se zpravidla měří 3 hodnoty intenzit v odpovídajících částem barevného spektra. Výsledný obraz je pak složen ze tří barevných kanálů – červeného, zeleného a modrého kanálu. U takového obrazu pak na každý pixel připadají 3 číselné hodnoty. Tato práce se však zabývá především obrazy formátu grayscale, a proto v dalších částech, pokud nebude řečeno jinak, může čtenář předpokládat, že se mluví o obraze ve formátu grayscale. Kvalitu obrazu z pohledu člověka však především ovlivňuje počet úrovní, na které je amplituda obrazové funkce kvantována. Při malém množství úrovní totiž dochází ke vzniku falešných obrysů. Například pokud je intenzita každého pixelu vyjádřena pomocí 8 bitů, pak je obrazová funkce kvantována do 256 úrovní. V praxi je tento počet bitů nejčastější,

ale používají se samozřejmě jak menší, tak větší počty bitů pro kvantování [2, s. 9-13]. Proces kvantování je zachycen na obr. 1. Na tomto obrázku lze vidět, že každý pixel obsahuje pouze jednu úroveň intenzity (jednu úroveň pro každý barevný kanál), která je průměrem oblasti pokryté daným pixelem.



Obr. 1: Obrázek ilustruje pořízení digitálního obrazu. Fotografie na pozadí přestavuje reálný svět a matice pixelů m x n představuje pořízený digitální obraz [Obr. A].

2.3. Vlastnosti digitálního obrazu

Digitální obraz můžeme chápat jako matici o velikosti m x n, kde každý prvek matice odpovídá jednomu pixelu. Pixel je zkratkou anglického sousloví „Picture element“, což v překladu znamená prvek obrazu. Zároveň je pixel nejmenší jednotkou v obraze. U tvaru pixelu klademe požadavek na to, aby síť vytvořená z pixelů souvisle vyplňovala plochu. Tuto podmínku splňuje rovnostranný trojúhelník, čtverec a pravidelný šestiúhelník. Nejintuitivnější z těchto tří je samozřejmě čtvercový pixel resp. čtvercová mřížka, která se také používá nejčastěji. Nevýhodou čtvercové mřížky ale je, že pixel sdílí stranu pouze se 4 ze svých 8 sousedních pixelů.

Další často zmiňovaná vlastnost digitálního obrazu je jeho rozlišení. Rozlišení u digitálního obrazu je často bráno jako množství pixelů obsažených v obraze. Pokud ale mluvíme o rozlišení, můžeme tím také myslet velikost detailů v obraze, které jsme schopni rozeznat. V takovém případě rozlišení závisí na tom, jak velkou část obrazu zachycuje jeden pixel[1, s. 115].

3. Hrana v digitálním obraze

Hranou u digitálního obrazu rozumíme jeho část, ve které dochází k výrazné změně obrazové funkce $f(x, y)$. Příčinou vzniku hrany v obraze může být například to, že se v této části obrazu setkávají dva různé objekty, které se liší barvou, texturou, odrazivostí, osvětlením atd. Tedy hrana v obraze vzniká jako důsledek fyzických změn ve scéně zachycené obrazem. Matematicky jsou popsány vektorovou veličinou $gradient - \nabla$. Modul a směr gradientu můžeme vyjádřit následovně

$$|\nabla f(x, y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{3.1}$$

$$\theta = arctan\left(\frac{g_y}{g_x}\right), \tag{3.2}$$

kde $g_y = \partial f / \partial x$. V některých případech nás ovšem zajímá pouze velikost gradientu nezávisle na jeho směru. V takovém případě je jednodušší spočítat takzvaný Laplacián. Jedná se v podstatě o druhou mocninu gradientu. Jeho matematický zápis je následovný

$$\nabla^2 g(x, y) = \frac{\partial^2 g(x, y)}{\partial x^2} + \frac{\partial^2 g(x, y)}{\partial y^2} \text{ [1, s. 163-4].} \tag{3.3}$$

3.1. Korelační a konvoluční operátory

Při detekci hran a obecně při zpracování obrazu využíváme matematických nástrojů jako je derivace, parciální derivace atd. Při aplikaci na diskrétní data ale tyto nástroje aproximujeme lineárními operátory. Mezi tyto operátory patří především korelace a konvoluce. Než budeme dále tyto nástroje a jejich použití rozebírat, zavedeme si pojem lineární operátor.

Nechť D je operátor, který aplikujeme na obraz f, výsledkem pak je obraz g. Pokud pro D platí

$$D(\alpha f_1 + \beta f_2) = \alpha D(f_1) + \beta D(f_2), \tag{3.4}$$

kde f_1 a f_2 jsou obrazy, α a β jsou skalární koeficienty, pak D nazýváme „lineární operátor“.

Pro začátek si demonstrujeme na příkladu aplikaci operátoru v jednorozměrném případě. Máme tedy jednorozměrnou masku h o třech prvcích a jednorozměrný obraz f o pěti prvcích.

h_{-1}	h_0	h_1

f_1	f_2	f_3	f_4	f_5

(3.5)

Korelací masky s obrazem dostaneme $g_3 = f_2 h_{-1} + f_3 h_0 + f_4 h_1$. Pokud bychom tento postup vyjádřili obecně pro dvourozměrný případ, dostali bychom následující vzorec

$$g(x,y) = \sum_{\alpha} \sum_{\beta} f(x + \alpha, y + \beta)\, h(\alpha, \beta) \tag{3.6}$$

Často se pak v praxi používá maska o rozměrech 3 x 3 a α, β nabývají hodnot -1,0,1. Konvoluce je korelaci velice podobná, což je na první pohled vidět ze vzorce pro obecnou konvoluci ve dvoudimenzionálním prostoru

$$g(x,y) = \sum_{\alpha} \sum_{\beta} f(x - \alpha, y - \beta)\, h(\alpha, \beta). \tag{3.7}$$

Konvoluce se tedy liší pouze tím, že na rozdíl od korelace nenásobíme např. levý horní prvek masky s odpovídajícím bodem obrazu (také levý horní), ale násobíme ho s pravým dolním bodem [3, s. 65-7].

3.2. Využití jader pro odhad derivací

Nejdříve si připomeňme, jak je definována parciální derivace.

$$\frac{\partial f}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}. \tag{3.8}$$

Protože se jedná o digitální obraz, který je diskrétní, nejmenší možný krok je 1 pixel, a tudíž se $\Delta x = 1$. Pokud bychom tedy do rovnice (2.7) dosadili, dostali bychom následující jádro (v anglické literatuře „kernel“)

-1	1

(3.9)

Při aplikaci tohoto operátoru na bod obrazu $f(x,y)$, pro který derivaci odhadujeme, se nad tímto bodem bude nacházet prvek jádra obsahující hodnotu -1. Hned na první pohled

je jasné, že jádro (2.5) je nesymetrické. Tato nesymetrie způsobí, že bereme v úvahu pouze rozdíl mezi zkoumaným pixelem a jeho sousedem vpravo. Existuje samozřejmě i symetrická varianta, jejíž rovnice vypadá takto

$$\left.\frac{\partial f}{\partial x}\right|_{x_0} = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0 - \Delta x)}{2\Delta x}. \tag{3.10}$$

Do rovnice (2.9) dosadíme podobně, jako jsme to udělali u rovnice (2.7). Výsledkem je následující jádro

$$1/2 \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline \end{array} \tag{3.11}$$

Výhodou tohoto jádra je jeho symetrie. Naopak nevýhodou je, že nebereme v potaz hodnotu pixelu, na který operátor aplikujeme. Dále je potřeba ošetřit vliv šumu, který je derivací zesílen. Abychom tento vliv snížili, většinou požíváme jádro se 3 řádky, u kterého se v každém řádku spočítá diference, a výsledek je pak dán průměrem těchto tří řádků. Průměrováním tak dosáhneme omezení vlivu šumu. Takovéto jádro vypadá následovně

$$1/6 \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \tag{3.12}$$

Podobně bychom mohli navrhnout jádro, které by aproximovalo druhou derivaci, nejrůznější druhy filtrů, nebo také jejich kombinace [3, s. 67-8].

3.3. Detekce hran

Přístupů využívaných při detekci hran existuje mnoho. Většina z nich se snaží vytvořit nový, nejčastěji binární obraz, kde buď pixel náleží dané hraně, nebo jí nenáleží. V této práci se zaměřím podrobněji na hranové detektory, které využívají k vytvoření hranového obrazu lineární operátory, resp. jádra, o kterých byla řeč v přechozí kapitole. Samozřejmě, že existují i další metody detekce hran – např. detekce hran pomocí morfologických filtrů nebo jiných nelineárních operátorů. Tyto metody však přesahují rámec této práce, a proto se budu dále věnovat pouze metodám detekce hran založených na aplikaci lineárních operátorů. Tyto metody se dají klasifikovat do 3 kategorií:

1. Jako první se rozvinuly metody, které pomocí lineárních operátorů aproximují první derivaci a většinou se zároveň chovají jako filtry s dolní propustí, které vyhlazují obraz. Aproximace parciálních derivací $f_x = \partial f / \partial x$ a $f_y = \partial f / \partial y$ vznikne konvolucí obrazu s malým jádrem, typicky se jedná o jádro s rozměry 3 x 3. Poté nelineární kombinací těchto dvou aproximací f_x a f_y dostaneme modul gradientu $|\nabla f(x, y)|$. Produktem této metody je tedy nový obraz, kde hodnota každého pixelu odpovídá modulu gradientu a kde se v místě hrany nachází lokální maximum obrazové funkce $f(x, y)$. Hrany jsou pak většinou identifikovány jednoduchým prahováním nově vzniklého obrazu. Příkladem takovýchto hranových detektorů, resp. operátorů jsou: operátor Prewittové, Sobelův nebo také Kirschův operátor. Podívejme se tedy, jak vypadají jádra těchto zmiňovaných operátorů [2, s. 310].

Operátor Prewittové

$$h_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad h_y = \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array} \tag{3.13}$$

Sobelův operátor

$$h_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad h_y = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array} \tag{3.14}$$

Kirschův operátor

$$h_x = \begin{array}{|c|c|c|} \hline -5 & 3 & 3 \\ \hline -5 & 0 & 3 \\ \hline -5 & 3 & 3 \\ \hline \end{array} \qquad h_y = \begin{array}{|c|c|c|} \hline 3 & 3 & 3 \\ \hline 3 & 0 & 3 \\ \hline -5 & -5 & -5 \\ \hline \end{array} \tag{3.15}$$

Problémem těchto detektorů je jednak citlivost na šum, ale především také to, že neberou v potaz měřítko významných objektů, které se v obraze nacházejí. To spolu s jednoduchým prahováním může mít za následek, že nedetekujeme detaily,

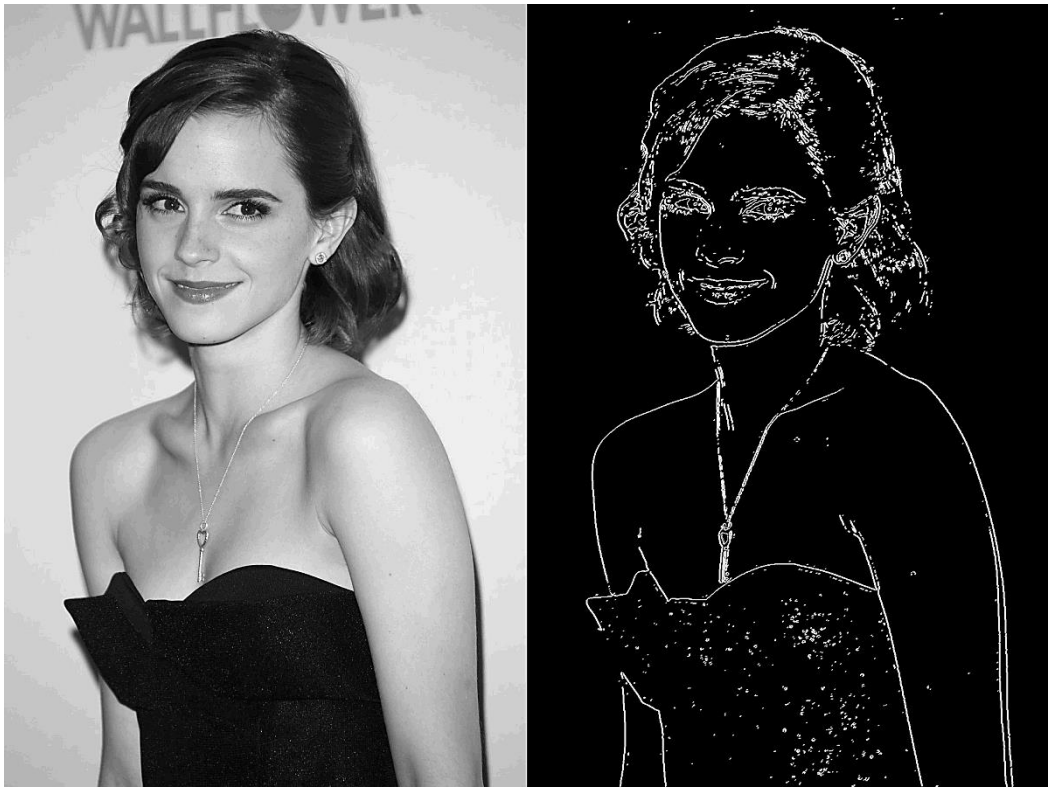
které jsme detekovat chtěli [1, s. 165-7]. Detekce hran pomocí operátoru Prewittové viz. obr. 2.

Obr. 2: Vlevo původní fotografie, uprostřed modul gradientu a vpravo detekované hrany. Pro detekci hran byl použit operátor Prewittové [Obr. B].

2. Další metody, které se rozvinuly, detekovaly hrany jako místa, kde druhá derivace obrazové funkce prochází nulou. Už v předchozím bodě, kde se mluvilo o aproximaci první derivace, byl zmíněn problém se šumem. Pokud bychom tedy principiálně stejným způsobem aproximovaly druhou derivaci, vliv šumu by byl už příliš vysoký. Elegantní způsob, ne však zcela optimální, se podařilo najít Marrovi a Hildrethové, kteří se ve své teorii inspirovali u živých tvorů. Aby zmírnili vliv šumu, obraz nejdříve vyhladili pomocí Gaussovy funkce

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} . \tag{3.16}$$

Pro odhad druhé derivace použili Marr s Hildrethovou Laplacián, který byl zmíněn v rovnici (2.3). Protože jsou tyto dvě výše zmíněné operace lineární, lze jejich pořadí zaměnit a provést jako konvoluci s jediným jádrem. Operátor spojující aplikaci Laplaciánu a filtraci pomocí Gaussovy funkce se jmenuje Laplacián Gaussiánu - $\nabla^2 G$ (v anglické literatuře – Laplacian of Gaussian) [2, s. 310-1]. Detekce hran pomocí LoG je ilustrována na obr. 3.



Obr. 3: Příklad použití Laplaciánu Gaussiánu pro detekci hran v obraze. Parametr $\sigma = 2$ [Obr. B].

3. Třetí skupinu reprezentuje Cannyho hranový detektor. John F. Canny v 80. letech řešil ještě jako doktorand problém detekce skokových změn v 1D signálu [7]. Snažil se najít optimální řešení pro tento problém. Jeho metoda je založená na myšlence, že detekci skokových změn zašuměného signálu lze provést pomocí aplikace filtru na signál. Hrany byly detekovány jako maxima obrazu vzniklého lineární konvolucí původního obrazu a filtru s konečnou impulzní odezvou. Cannyho hranový detektor se snaží především o maximalizaci těchto tři stěžejní parametrů: kvalitní a robustní detekci vzhledem k šumu, přesnou lokalizaci hrany a jedinečnost hrany pro její nejbližší okolí [2, s. 311].

   Prvním krokem při detekci hran Cannyho detektorem je vyhlazení obrazu. Filtr s optimální odezvou, který maximalizuje výše zmíněné parametry, lze aproximovat pomocí Gaussiánu. Gaussián pro 2D obraz je uveden v rovnici (2.15).

   Dalším krokem je nalezení přechodů nuly v obraze, na který byl aplikován diferenciální operátor 2. řádu. Výhodou oproti Laplaciánu Gaussiánu je, že

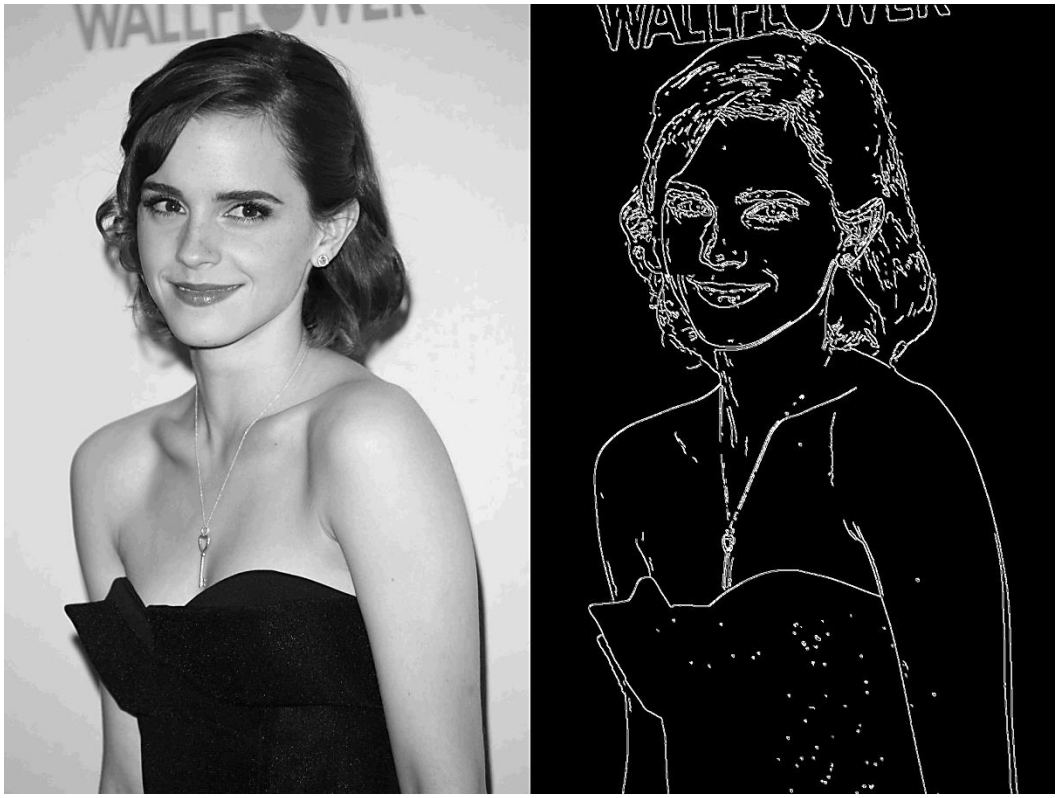
kromě modulu gradientu uvažujeme i směr hrany. Pro určení směru gradientu používáme jeho robustní odhad

$$\boldsymbol{n} = \frac{\nabla(G * f)}{|\nabla(G * f)|}, \tag{3.17}$$

kde $\boldsymbol{n}$ představuje směr gradientu, G je 2D Gaussián a f značí obraz, pro který směr gradient odhadujeme. Přesnou lokalizaci hrany ve směru gradientu vypočteme následovně

$$\frac{\boldsymbol{\partial^2}}{\boldsymbol{\partial n^2}} G * f = 0 . \tag{3.18}$$

Rovnice (2.17) představuje takzvanou metodu potlačení nemaximálních odezev (v angličtině non-maximum suppression). Tato metoda byla tedy navržena pro splnění podmínky na přesnou lokalizaci hrany a s tím spojený problém více odezev na jednu hranu. V rovnici (2.17) dochází současně k aplikaci Gaussiánu i derivace zároveň. V praxi je ale častější nejdříve „vyhladit“ obraz Gaussiánem a poté aplikovat derivaci.



Obr. 4: Příklad použití Cannyho hranového detektoru [Obr. B].

Posledním krokem algoritmu je vytvoření binárního obrazu, kdy je buď pixel prohlášen za hranu, nebo ne. Tohoto se docílí pomocí prahování s hysterezí, které používá na rozdíl od jednoduchého prahování dva prahy. Pixely, jejichž hodnota se nachází mezi těmito prahy, jsou prohlášeny za hranu, pokud se nacházejí v blízkosti pixelu s hodnotou vyšší než horní práh [1, s. 171-3]. Nakonec se ještě některé verze Cannyho detektoru pokouší spojit hrany tak, aby vznikly skutečně souvislé hrany.

3.4. Houghova transformace

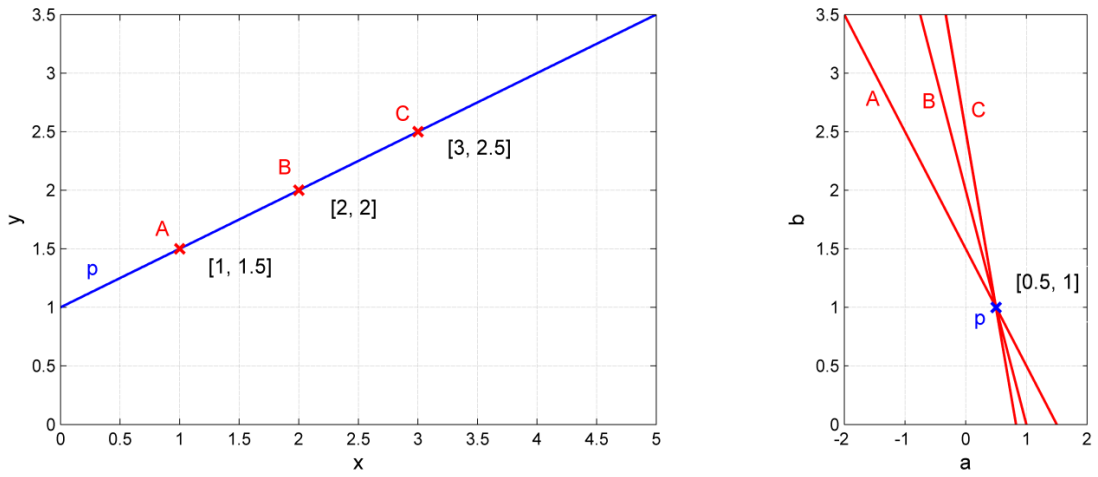
Metodami detekce hran, které byly rozebrány v předchozí kapitole, získáme nový hranový obraz. Takový obraz je ale pouze maticí dat, kde každý bod obsahuje binární informaci, zda se v daném místě nachází hrana. Často je pro další zpracování obrazu vhodné najít nějaký vhodný popis pro jednotlivé hrany. Například pokud je zapotřebí obraz segmentovat, nebo v něm vyhledat některé význačné body. Existují různé přístupy k tomuto problému, některé metody například spojují shluky pixelů, které na sebe navazují, do spojitých linií a pak s těmito objekty dále pracují. Existují také metody, které se snaží útvary vzniklé v hranovém obraze popsat matematickými výrazy. Právě mezi tyto metody patří Houghova transformace, jejíž původní verze byla navržena pro hledání přímek v hranovém obraze. Uvažujme tedy rovnici přímky

$$y = ax + b, \tag{3.19}$$

kde a, b jsou parametry popisující danou přímku. Pak rovnici pro každý bod této přímky můžeme přepsat do tvaru

$$b = y - ax, \tag{3.20}$$

kde x, y považujeme za konstanty a a, b za parametry. Toto nazýváme parametrickou transformací. Každému bodu obrazu po transformaci do prostoru parametrů $b - a$ odpovídá přímka [3, s. 275-6].



Obr. 5: Transformace mezi obrazem a prostorem parametrů.

Na obrázku č. 5 je ilustrována klíčová vlastnost této transformace - totiž, že body obrazu ležící na stejné přímce se v prostoru parametrů zobrazí jako přímky, které mají průsečík právě v bodě, jehož souřadnice odpovídají parametrům přímky z původního obrazu. Pokud tuto transformaci aplikujeme na všechny body daného obrazu, vznikne nám nový obraz, ve kterém můžeme nalézt přímky jako lokální maxima obrazové funkce $f(a, b)$. Tento postup si také můžeme představit jako proces, při kterém každý bod z původního obrazu odevzdá hlas pro několik variant. Varianty (resp. parametry přímek), které získají nejvíce hlasů, jsou považovány za výsledek [5, s. 218-9].

Houghova transformace byla později zobecněna pro nalezení libovolných tvarů, nejčastěji kružnic a elips. Autory zobecnění jsou Richard Duda a Peter Hart [9].

4. Projektivní transformace ve 2D

Projektivní geometrie je matematický obor zabývající se geometrickými vlastnostmi, které jsou invariantní vůči projektivním transformacím. Tato geometrie se začala prudce rozvíjet v 19. století a našla široké uplatnění zejména v počítačovém vidění a grafice. Projektivní geometrie zavádí několik pojmů, se kterými se v klasické Eukleidovské geometrii nesetkáváme. Například prostor projektivní geometrie obsahuje oproti Eukleidovskému navíc body v nekonečnu. Zároveň platí, že každé dvě přímky mají průsečík, a to i rovnoběžné přímky, které se protínají právě v nekonečnu. Projektivní geometrie pracuje především s body, přímkami a plochami. Naopak ale nevyužívá nástroje, resp. pojmy, jako jsou např. úhel nebo vzdálenost. Tyto pojmy totiž nejsou invariantní vůči projektivním transformacím. Dalším důležitým pojmem je princip duality. Tento princip ukazuje, že je možné ve všech teorémech týkajících se projektivní geometrie v prostoru 2D zaměnit pojmy přímka a bod [6, s. 25-30].

V projektivní geometrii se také využívají homogenní souřadnice, které mají tu vlastnost, že dokáží vyjádřit souřadnice bodu v nekonečnu pomocí reálných čísel. Homogenní souřadnice mají na rozdíl od Kartézských souřadnic vždy o jednu souřadnici více než je dimenze daného prostoru. Dále pro ně platí, že dva body v projektivním prostoru jsou identické, pokud souřadnice jednoho bodu jsou skalárním násobkem souřadnic bodu druhého. Matematicky bychom tuto skutečnost vyjádřili následovně

$$(x_1, y_1, w_1) = (kx_2, ky_2, kw_2). \tag{4.1}$$

Díky tomu lze vyjádřit projektivní transformace pomocí matic, což je samozřejmě důležité pro využití v počítačové technice.

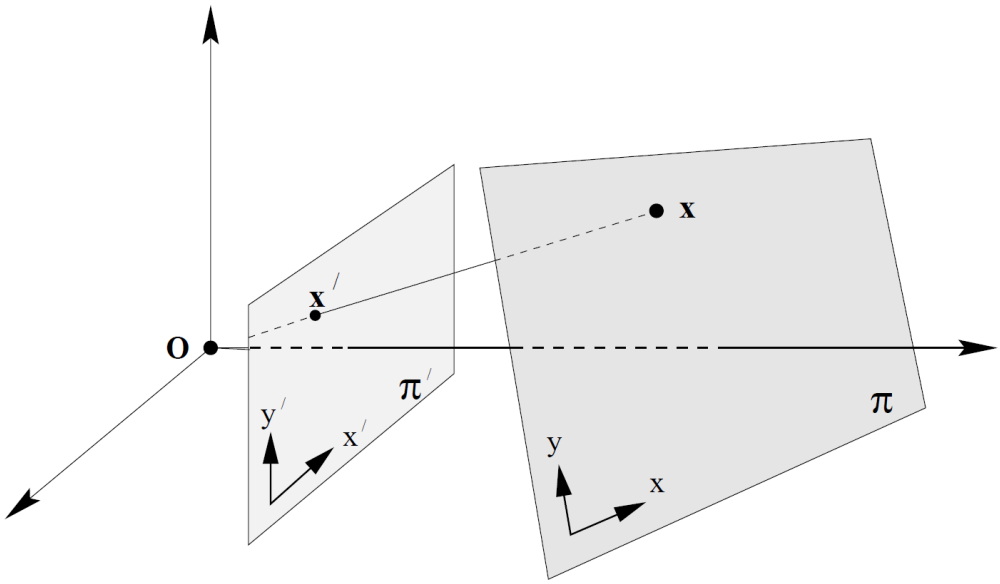
4.1. Homografie ve 2D

Projektivní transformace ve dvoudimenzionálním prostoru je lineární transformace homogenního tříprvkového vektoru nesingulární maticí velikosti 3 x 3. V maticovém zápisu bude tato transformace vypadat následovně

$$\boldsymbol{x}' = H\boldsymbol{x} \tag{4.2}$$

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \tag{4.3}$$

kde x_i jsou souřadnice transformovaného bodu a x_i' souřadnice nového bodu, získaného transformací, oboje v homogenních souřadnicích. Při bližším pohledu na matici H si můžeme všimnout, že pokud ji vynásobíme celou skalárem k, pak transformace zůstane nezměněna. Jediným rozdílem bude, že výsledný bod získaný transformací bude mít souřadnice k-krát větší, či menší. Jak již bylo ale výše zmíněno, u homogenních souřadnic je důležitý jejich poměr a ten zůstane zachován. Matici H proto nazýváme homogenní maticí.



Obr. 6: Projektivní transformace ve 2D neboli 2D homografie [obr. C, s. 34].

Matice koeficientů H tedy obsahuje 9 prvků, ale má pouze 8 stupňů volnosti, protože je nezávislá na měřítku. Můžeme si tedy položit otázku, kolik je potřeba sobě si odpovídajících dvojic bodů $\boldsymbol{x}' \leftrightarrow \boldsymbol{x}$, abychom získali řešení. S každou dvojicí bodů $\boldsymbol{x}', \boldsymbol{x}$ získáme 2 vazební podmínky. Pro získání řešení jsou tedy potřeba 4 dvojice odpovídajících bodů. Rovnici (4.2) lze také přepsat do tvaru

$$\boldsymbol{x}_i' \times H\boldsymbol{x}_i = 0. \tag{4.4}$$

Dále můžeme rozepsat $\boldsymbol{x}_i'$ jako $(x_i', y_i', w_i')^{\mathrm{T}}$ a vyjádřit výše uvedené vektorové násobení jako

$$\boldsymbol{x}_i' \times H\boldsymbol{x}_i = \begin{pmatrix} y_i'\mathbf{h}^{\mathbf{3T}}x_i - w_i'\mathbf{h}^{\mathbf{2T}}x_i \\ w_i'\mathbf{h}^{\mathbf{1T}}x_i - x_i\mathbf{h}^{\mathbf{3T}}x_i \\ x_i\mathbf{h}^{\mathbf{2T}}x_i - y_i'\mathbf{h}^{\mathbf{1T}}x_i \end{pmatrix}, \tag{4.5}$$

kde $\mathbf{h}^{\mathbf{jT}}$ značí j-tý transponovaný řádek matice H. Výše uvedený výraz můžeme dále upravit, abychom měli zvlášť v jedné matici koeficienty a v druhé hledaný vektor řešení.

$$A_i\mathbf{h} = \begin{pmatrix} \mathbf{0}^{\mathrm{T}} & -w_i'\boldsymbol{x}_i^{\mathbf{T}} & y_i'\boldsymbol{x}_i^{\mathbf{T}} \\ w_i'\boldsymbol{x}_i^{\mathbf{T}} & \mathbf{0}^{\mathrm{T}} & -x_i\boldsymbol{x}_i^{\mathbf{T}} \\ x_i\boldsymbol{x}_i^{\mathbf{T}} & -y_i'\boldsymbol{x}_i^{\mathbf{T}} & \mathbf{0}^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} \mathbf{h}^{\mathbf{1}} \\ \mathbf{h}^{\mathbf{2}} \\ \mathbf{h}^{\mathbf{3}} \end{pmatrix}, \tag{4.6}$$

V rovnici (4.6) si může pozorný čtenář všimnout, že první dva řádky matice A_i jsou lineárními kombinacemi třetího řádku. Třetí řádek můžeme tedy zcela vypustit a výsledný vztah poté vypadá následovně

$$A_i\mathbf{h} = \begin{pmatrix} \mathbf{0}^{\mathrm{T}} & -w_i'\boldsymbol{x}_i^{\mathbf{T}} & y_i'\boldsymbol{x}_i^{\mathbf{T}} \\ w_i'\boldsymbol{x}_i^{\mathbf{T}} & \mathbf{0}^{\mathrm{T}} & -x_i\boldsymbol{x}_i^{\mathbf{T}} \end{pmatrix} \begin{pmatrix} \mathbf{h}^{\mathbf{1}} \\ \mathbf{h}^{\mathbf{2}} \\ \mathbf{h}^{\mathbf{3}} \end{pmatrix}. \tag{4.7}$$

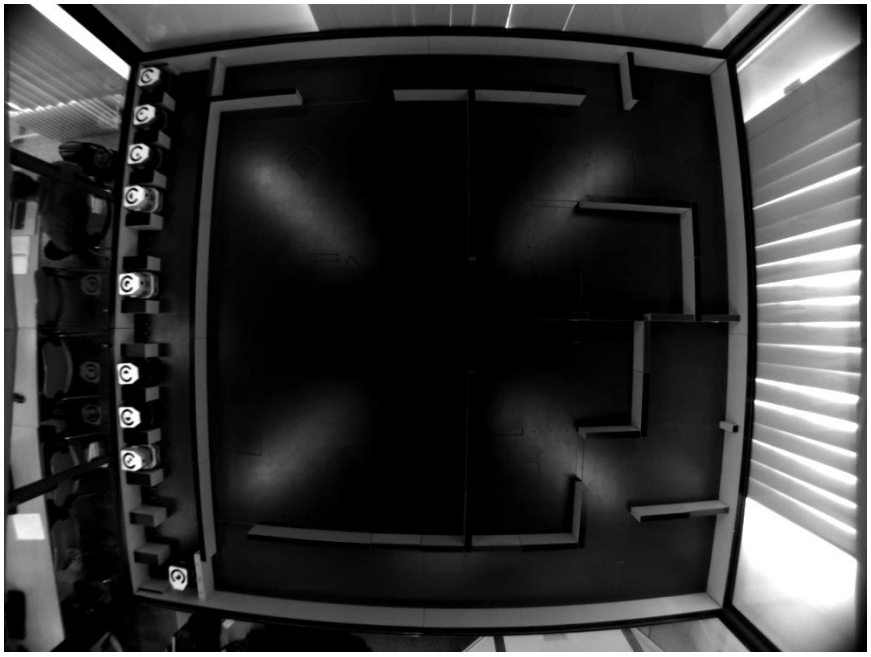
Matice A_i má velikost 2 x 9, což odpovídá jedné dvojici bodů $\boldsymbol{x}' \leftrightarrow \boldsymbol{x}$. Jak bylo ale už výše zmíněno, pro získání řešení je zapotřebí alespoň 4 dvojic bodů. Pro takovou soustavu rovnic má pak matice koeficientů A velikost 8 x 9 [6, s. 87-9].

5. Návrh a implementace programu

V této části se zaměřím na řešení praktické úlohy, tedy kalibrace vizuálního lokalizačního zařízení. Nejprve popíši konkrétní podmínky a parametry řešené úlohy. Dále budu psát o implementaci první části úlohy, ve které se snažím detekovat hrany v obraze získaném z kamery. Součástí této podkapitoly také budou další operace s hranovým obrazem důležité pro následné hledání rohů hřiště. Problému nalezení rohů hřiště bude věnována další podkapitola. V poslední části této kapitoly se budu věnovat získání transformační matice pomocí nalezených rohů hřiště.

5.1. Parametry úlohy

Úkolem bylo navrhnout a implementovat metodu kalibrace transformace mezi souřadnicemi v obraze a v pracovní ploše (aréna, ve které se roboty pohybují). Tato metoda, resp. program, má být začleněna do stávající systému. Program jsem psal v jazyce C++, ve kterém jsou napsány i metody stávajícího systému. Vstupem do mého programu je matice dat reprezentující obraz zachycený kamerou Unibrain Fire-i 820b, která je umístěna nad hřištěm [8, s. 8]. Rozlišení této kamery je 1200 x 1600 pixelů. Kamera nepracuje s jednotlivými barevnými kanály, ale pouze s celým barevným spektrem, tzn. výstupem z kamery je obraz ve stupních šedi (grayscale). Příklad obrazu, pořízeného kamerou za dobrých světelných podmínek, je vidět na Obr. 7.



Obr. 7: Snímek hřiště pořízený kamerou Unibrain Fire-i 820b. Obrázek byl dodatečně zesvětlen.

Úkolem programu je v obraze identifikovat rohy hřiště a poté ze znalosti polohy těchto bodů jak v obraze, tak v reálném prostředí, spočítat transformaci. Vypočtení transformace by mělo být dostatečně robustní, aby i v případě, že je v provozu pouze umělé osvětlení nad hřištěm arény, bylo zaručeno spolehlivé a přesné určení transformace. Přesnost stávajícího určení polohy robotů v aréně je $\pm$ 3 mm [8, s. 8]. Samozřejmě snahou je co nejvíce se této přesnosti přiblížit. Předpokládá se zároveň, že určování transformace nepoběží v reálném čase. Transformace pravděpodobně bude získávána při startu celého systému, a poté v určených časových intervalech, např. 1 hodina, jako kontrola platnosti této transformace. V současné podobě je transformace počítána staticky, to znamená, že při změně polohy kamery přestane daná transformace platit. Pokud takový problém nastane, je nutné, aby správce systému SyRoTek manuálně nastavil kameru zpátky do správné polohy. Mnou implementované řešení by při správné funkčnosti mělo tedy takovéto zásahy výrazně omezit.

5.2. Detekce hran

Po zvážení několika variant jsem se rozhodl pro implementaci hranových detektorů založených na konvolučních operátorech. Tato problematika byla podrobněji rozebrána v kapitole 3, konkrétně pak v podkapitole 3.3. Jako první jsem tedy implementoval detektor využívající operátor Prewittové. Na tomto detektoru jsem si především vyzkoušel práci se samotným obrazem, konkrétně načtení dat z obrazu do matice a zpětné ukládání dat jako obrazový soubor. Dále jsem napsal funkci pro obecnou konvoluci ve 2D, která jsem později využil i u dalších hranových detektorů. Výsledek byl při ideálním osvětlení dobrý, ale při horším osvětlení, tzn. pokud bylo po setmění a v provozu bylo pouze osvětlení samotné arény, byl výsledný hranový obraz nedostačující.

Jako další jsem implementoval detektor Laplacián z Gaussiánu (zkráceně LoG). Při jeho programování jsem využil konvoluční masku tvaru tzv. „mexického klobouku“ o velikosti 5 x 5, uvedenou v [1, s. 169]. Hrany jsem pak nalezl jako přechody nuly. Výsledek byl o něco lepší než u operátoru Prewittové, i přesto jsem se ale rozhodl zkusit implementovat další, o něco složitější, ale zato podle literatury kvalitnější detektor.

Cannyho hranový detektor

Pro implementaci Cannyho hranového detektoru jsem zvolil mírně zjednodušenou verzi. Zjednodušení spočívá především v tom, že pro získání derivací používám podobné jádro jako je u operátoru Prewittové a pro získání úhlu gradientu využívám jeho modul ve

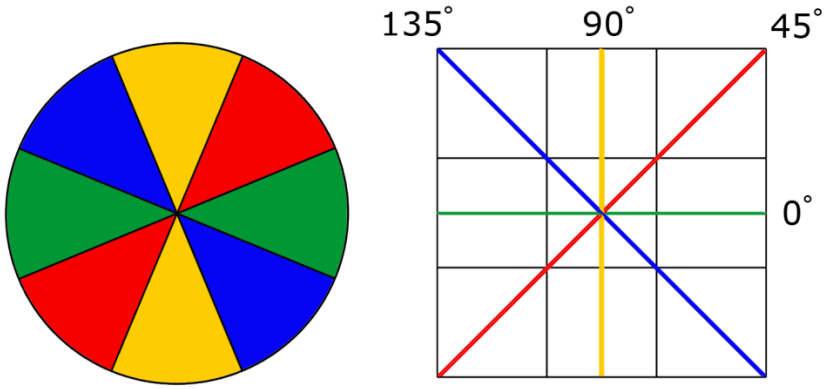
směru osy x a osy y. Tato verze se ale osvědčila a výsledek byl téměř stejný, jako při použití funkce z knihovny programu Matlab. Průběh hledání hran Cannyho detektorem se dá rozdělit do čtyř následujících kroků:

Krok 1: Odstranění šumu resp. vyhlazení obrazu pomocí Gaussova filtru. Pro tento krok jsem použil následující masku o rozměrech 5 x 5.

$$1/159 \quad \begin{array}{|c|c|c|c|c|} \hline 2 & 4 & 5 & 4 & 2 \\ \hline 4 & 9 & 12 & 9 & 4 \\ \hline 5 & 12 & 15 & 12 & 5 \\ \hline 4 & 9 & 12 & 9 & 4 \\ \hline 2 & 4 & 5 & 4 & 2 \\ \hline \end{array} \tag{5.1}$$

Výše uvedená maska aproximuje Gaussův filtr s parametrem $\sigma = 1.4$.

Krok 2: Výpočet modulu a úhlu gradientu. Výstupem tohoto kroku jsou dva nové obrazy. První obraz obsahuje modul gradientu, vypočtený konvolucí s operátorem aproximujícím derivaci. Druhý obsahuje číslo 1 až 4, podle toho, jaký je směr gradientu v daném místě. Rozdělení do čtyř základních směrů je znázorněno na obr. 8.

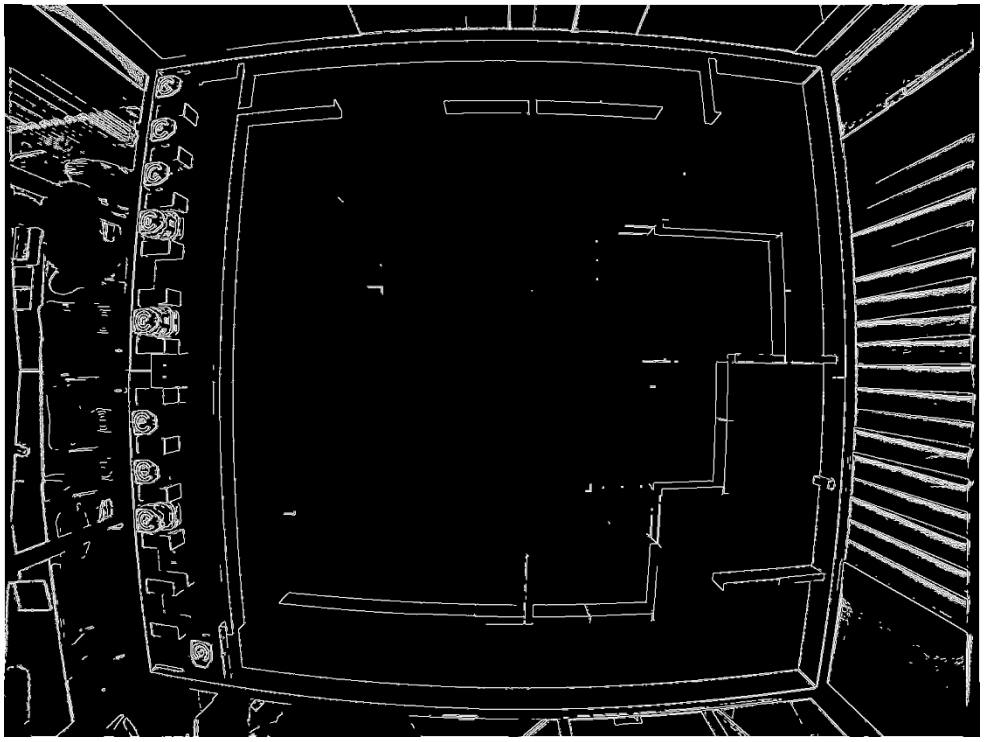


Obr. 8: Ilustrace rozdělení úhlů gradientu do čtyř skupin. Rozdělení souvisí s tzv. 8-okolím pixelu, což je ilustrováno obrázkem vpravo.

Krok 3: Potlačení nemaximálních hran. V této části algoritmu se za pomocí modulu a úhlu gradientu zjišťuje, zda se v okolí pixelu ve směru gradientu nenachází pixel s vyšší hodnotou modulu gradientu. Pokud takový pixel existuje, pak je hodnota původního pixelu potlačena, resp. je potlačena hodnota modulu gradientu. Podle toho do jaké kruhové výseče patří úhel gradientu, se prohledává odpovídající okolí pixelu.

Krok 4: Prahování s hysterezí využívající dvou prahů. Pokud je hodnota modulu gradientu nižší, než je nižší z prahů, pak je pixel okamžitě zamítnut. Pokud je hodnota modulu gradientu naopak vyšší než vyšší z prahů, pak je tento pixel rovnou považován za součást hrany. U pixelů, jejichž hodnota se nachází mezi těmito prahy, se pak zkoumá jejich okolí. Pokud navazují na jiný hranový pixel, pak se z nich také stane součást hrany. Tento krok má obecně za cíl, aby výstupem celého Cannyho detektoru byly propojené hranové útvary.

Výsledek Cannyho detektoru byl znovu závislý na osvětlení jako v předchozích případech, ale výsledky Cannyho detektoru byly ze všech vyzkoušených detektorů nejlepší.

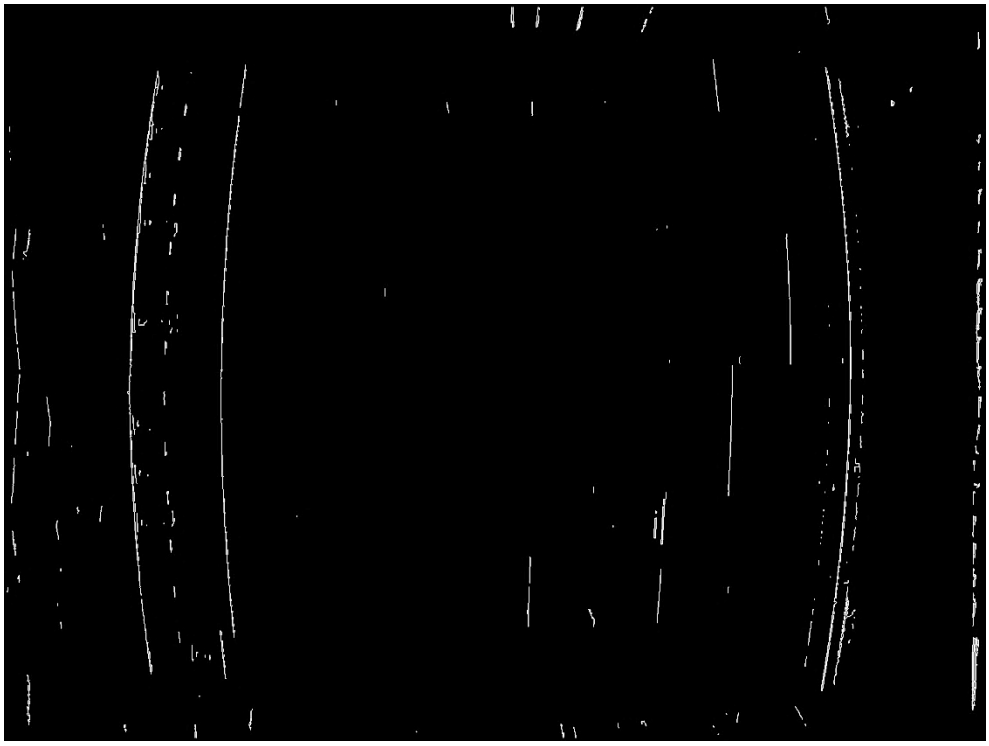


Obr. 9: Hrany detekované Cannyho detektorem.

5.3. Další zpracování detekovaných hran

Na obr. 9 lze vidět, že množství detekovaných hran je příliš velké a že obvodová hrana hřiště, kterou potřebuji nalézt, se v tomto množství ztrácí. Otázkou tedy bylo, jak množství hran omezit, ale přitom nepřijít o obvodovou hranu. Při pohledu na obrázek z kamery (viz. Obr. 7) jsem si všiml, že pro hledanou obvodovou hranu vždy platí, že tvoří rozhraní mezi vnitřními světlými plochami a vnějšími tmavými plochami. Tohoto zjištění jsem využil a navrhl funkci, která vezme okolí hranového pixelu a rozdělí ho na

dvě poloviny podle směru gradientu. Přímka kolmá na směr gradientu tvoří hranici mezi těmito dvěma polovinami. U takto získaných polovin okolí pixelu jsou sečteny hodnoty jednotlivých pixelů každé ze dvou polovin, čímž získám dvě hodnoty. Tyto hodnoty využiji k určení strany, která je světlejší. S využitím dalších informací, např. o jaký kvadrant obrazu se jedná, pak rozhodnu, zda daný hranový pixel v obraze ponechám, nebo ne. Výstup z této funkce pro vertikální hrany je vidět na obr. 10.

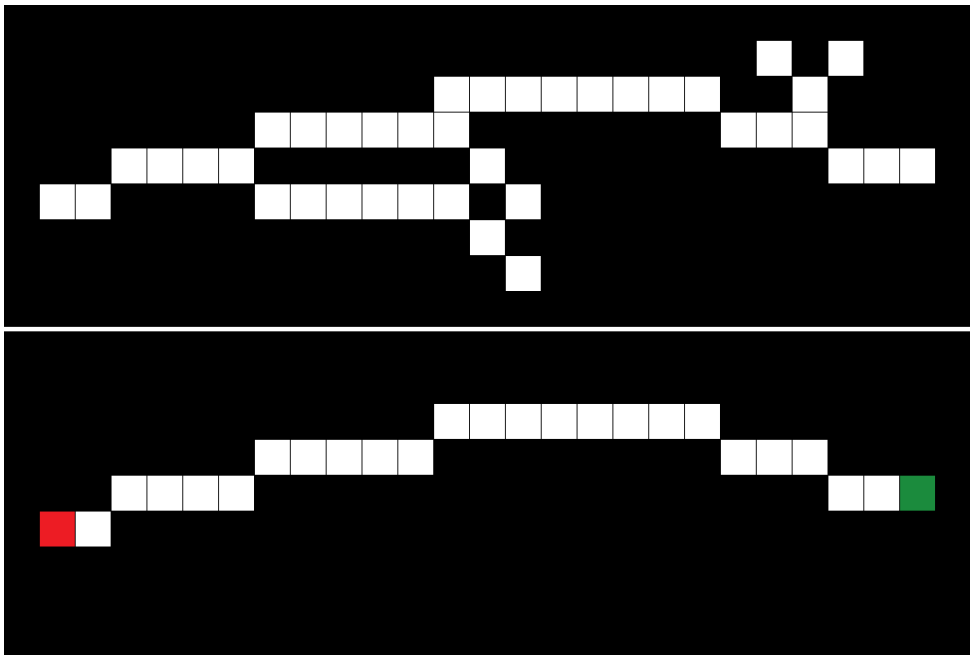


Obr. 10: Vybrané vertikální hranové pixely.

Jak lze vidět, funkce výrazně omezila nebo alespoň roztříštila přebytečné hrany, které nehledám. V ideálním případě by funkce ponechala hledanou obvodovou hranu nepřerušovanou linií, ve skutečnosti se ale mohou v obraze vyskytovat místa, kde funkce obvodovou hranu přeruší. Z tohoto důvodu následuje další funkce, která má za úkol takto přerušené hrany znovu propojit. K propojení hran dochází mezi dvěma hranovými pixely, které sousedí právě s jedním hranovým pixelem (tzv. weak pixels) a které jsou zároveň dostatečně blízko sebe, tzn. 4 – 6 pixelů vzdálené. Samozřejmě takové propojení může dát vzniknout uzavřeným smyčkám v hranovém obraze. S tím je ale počítáno a je to v dalších fázích běhu programu ošetřeno.

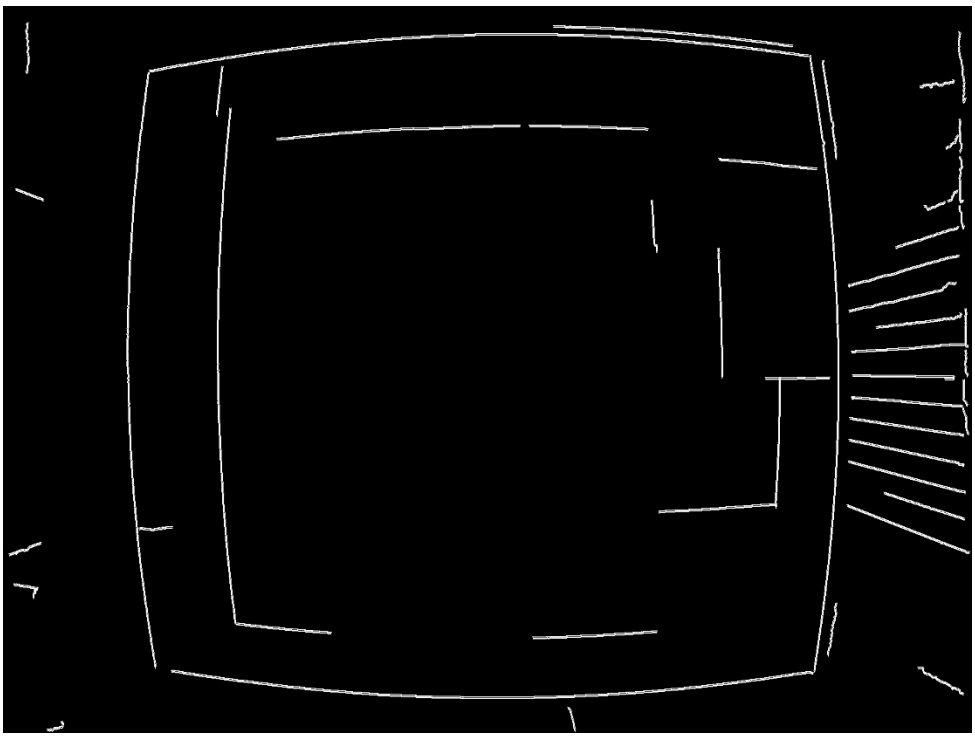
5.4. Tvorba objektů z hranových pixelů

Pro další práci s hranami připadalo v úvahu několik možností včetně Houghovy transformace, která byla popsána v podkapitole 3.4.. Tato metoda má své nesporné výhody, mezi ně patří například její univerzálnost, především pak u zobecněné Houghovy transformace. Samozřejmě má tato metoda i své nevýhody, např. její složitost se exponenciálně zvyšuje s počtem rozměrů potřebných k popisu hledaného matematického útvaru. Nakonec jsem se rozhodl vytvořit následující řešení, které sice není tak univerzálně využitelné, zato ale odpovídá více nárokům řešeného problému. Rozhodl jsem se implementovat metody, pomocí nichž bych z na sebe navazujících hranových pixelů vytvořil objekty – hrany. Prvním dílčím krokem je nalezení shluku sousedících pixelů (při prohledávání okolí pixelu se vždy pracuje s celým „8-okolím“). Takový shluk pixelů je ilustrován vrchní částí obr. 11. Jedná se o problém neinformovaného prohledávání stavového prostoru, u kterého neznáme počáteční a koncový bod prohledávání. Implementoval jsem tedy prohledávání do šířky (v angličtině – Breadth first search), které je pro tento problém vhodné. V dalším kroku jsem potřeboval odstranit zbytečné rozvětvení hrany tak, aby výsledný hranový objekt tvořil křivku, jak to vidíme např. v dolní části obr. 11. K tomu bylo zapotřebí nejdříve identifikovat body, které by sloužily jako start a cíl při dalším prohledávání. Protože primární funkcí je nalezení obvodových hran, což jsou dlouhé křivky, potřebuji nalézt cestu mezi dvěma nejvzdálenějšími body.



Obr. 11: Detail obrazu obsahujícího hranové pixely.

Tyto dva body poté prohlásím za start a cíl – v obr. 11 jsou to pixely označené zelenou a červenou barvou. Posledním krokem je pak nalezení nejkratší cesty mezi startovním a cílovým pixelem. Výstupem zmíněných funkcí je skupina pixelů (obr. 11 – dolní část), ze které je pak vytvořen nový hranový objekt.



Obr. 12: Hranové objekty vytvořené z binárního obrazu obsahujícího hranové pixely.

5.5. Identifikace obvodových hran arény

Z obr. 12 je patrné, že obvodové hrany se od ostatních liší především svou délkou. Jedná se o čtyři nejdelší hrany v obraze, zároveň jsou specifické svou vzájemnou polohou. Ještě než za pomoci těchto klíčových parametrů přistoupím k hledání obvodových hran, dochází k propojení blízkých hran. K propojování dochází zvlášť u horizontálně a vertikálně orientovaných hran. Vzdálenost těchto hran je zároveň také počítána s přihlédnutím na orientaci blízkých hran tak, aby například nedocházelo k propojování souběžných hran. V momentě, kdy má program k dispozici propojené hrany, najde čtyři největší a zkontroluje vzájemnou pozici. V případě, že kontrola proběhne úspěšně, jsou tyto čtyři hrany považovány za obvod hřiště.

5.6. Aproximace hran kružnicemi

K tomu, abych nalezl rohy hřiště, mi chyběl už jenom způsob, jakým bych vypočítal průsečíky obvodových hran. Jedním z možných přístupů by bylo čtyři hrany propojit a potom najít čtyři rohové pixely. Taková metoda by jistě ale postrádala potřebnou přesnost. Vzal jsem tedy každou ze čtyř hran a podíval jsem se na ni jako na část kružnice, pomocí které ji mohu aproximovat. Aproximaci kružnice provádím ve dvou krocích. V prvním kroku ze třech bodů hrany určím přibližnou polohu středu kružnice. Ve druhém kroku pak tuto polohu optimalizuji tak, aby vzdálenosti mezi jednotlivými body hrany a středem kružnice byly co nejméně rozdílné. Tedy dalo by se také říci, že se jedná o nalezení minimální odchylky, ve smyslu nejmenších čtverců. Poloha takto zjištěného středu je sice určena celými čísly (střed kružnice má formu pixelu), ale poloměr kružnic se pohybuje cca. kolem 3 tisíc pixelů, takže zaokrouhlení souřadnic na jeden celý pixel má v konečném výsledku jen zanedbatelný vliv.

Po aproximaci hran kružnicemi už zbývá jen výpočet jejich průniku. Obecně mohou mít dvě kružnice, resp. hrany až dva průniky, ale v mém případě se jen jeden z nich nachází uvnitř původního obrazu. Z těchto dvou průniků je tedy zvolen vždy ten, který se nachází v obraze, resp. ten který se nachází blíže středu obrazu. Získané čtyři průniky odpovídají souřadnicím rohů hřiště v obraze.

5.7. Získání transformace

Kamera, která snímá hřiště, snímá celé hřiště o rozměrech 3,5 x 3,8m ze vzdálenosti pouze cca. 2 metry [8, s. 7]. Kamera má tedy velký záběr, s tím souvisí i nepříjemnost v podobě soudkovitého zkreslení obrazu. Soudkovité zkreslení obrazu má za následek, že zvětšení se snižuje se vzdáleností od optické osy. Před vlastním výpočtem transformace je potřeba toto zkreslení odstranit. Je to z toho důvodu, že 2D homografie, jak byla popsána v kap. 4.1, je lineární transformace. Pokud by obraz zůstal zkreslený, vztah mezi obrazem a plochou hřiště by byl nelineární. Pro přepočet souřadnic ze zkresleného obrazu na souřadnice nezkresleného jsem využil zdrojový kód ze stávajícího systému. Závislost mezi těmito souřadnicemi se dá vyjádřit jako polynom, kde se koeficienty polynomu získají kombinací parametrů kamery a obrazu. V mém případě je potřeba získat souřadnice pouze čtyř bodů (rohů hřiště) v nezkresleném obraze.

V kap. 4.1 byla ze vztahu pro 2D homografii odvozena homogenní soustava lineárních rovnic, jejímž řešením je hledaná transformační matice. Pokud do vztahu (4.5) dosadíme pro čtyři dvojice bodů z obrazu a z plochy hřiště, dostaneme následující soustavu

$$A\mathbf{h} = \begin{pmatrix} \mathbf{0}^{\mathrm{T}} & -w_1'\boldsymbol{x}_1^{\mathbf{T}} & y_1'\boldsymbol{x}_1^{\mathbf{T}} \\ w_1'\boldsymbol{x}_1^{\mathbf{T}} & \mathbf{0}^{\mathrm{T}} & -x_1\boldsymbol{x}_1^{\mathbf{T}} \\ \mathbf{0}^{\mathrm{T}} & -w_2'\boldsymbol{x}_2^{\mathbf{T}} & y_2'\boldsymbol{x}_2^{\mathbf{T}} \\ w_2'\boldsymbol{x}_2^{\mathbf{T}} & \mathbf{0}^{\mathrm{T}} & -x_2\boldsymbol{x}_2^{\mathbf{T}} \\ \mathbf{0}^{\mathrm{T}} & -w_3'\boldsymbol{x}_3^{\mathbf{T}} & y_3'\boldsymbol{x}_3^{\mathbf{T}} \\ w_3'\boldsymbol{x}_3^{\mathbf{T}} & \mathbf{0}^{\mathrm{T}} & -x_3\boldsymbol{x}_3^{\mathbf{T}} \\ \mathbf{0}^{\mathrm{T}} & -w_4'\boldsymbol{x}_4^{\mathbf{T}} & y_4'\boldsymbol{x}_4^{\mathbf{T}} \\ w_4'\boldsymbol{x}_4^{\mathbf{T}} & \mathbf{0}^{\mathrm{T}} & -x_4\boldsymbol{x}_4^{\mathbf{T}} \end{pmatrix} \begin{pmatrix} \mathrm{h}^1 \\ \mathrm{h}^2 \\ \mathrm{h}^3 \\ \mathrm{h}^4 \\ \mathrm{h}^5 \\ \mathrm{h}^6 \\ \mathrm{h}^7 \\ \mathrm{h}^8 \\ \mathrm{h}^9 \end{pmatrix}, \tag{5.2}$$

kde $\boldsymbol{x}_i^{\mathbf{T}} = (x_i,\ y_i,\ w_i)$ a „nečárkované“ souřadnice jsou souřadnice bodů v obraze. V počítačové technice se pro nalezení řešení takovýchto soustav využívá rozklad na singulární hodnoty (v angličtině - Singular value decomposition, zkráceně SVD). Použitím SVD na matici A dostaneme

$$A = U\Sigma V^{\mathrm{T}}, \tag{5.3}$$

kde matice U, V jsou diagonální matice a Σ je ortogonální matice obsahující na diagonále singulární čísla. Pro můj problém je však především zajímavá matice V^{T}, která obsahuje tzv. pravé singulární vektory a její poslední sloupec obsahuje hodnoty blížící se hodnotám vektoru **h** [10, s. 1-2]. K výpočtu SVD jsem použil knihovnu GSL [12], která počítá SVD algoritmem publikovaným Golubem a Reinschem [11]. Jedním z důvodů, proč jsem si vybral tuto knihovnu, bylo, že na serveru SyRoTek už byla tato knihovna nainstalována.

Výpočtem SVD jsem tedy získal vektor řešení **h,** resp. jeho odhad s minimální odchylkou, počítanou metodou nejmenších čtverců. Výslednou matici transformace H, pro kterou bude platit $\boldsymbol{x}' = H\boldsymbol{x}$, získáme z vektoru **h** následovně

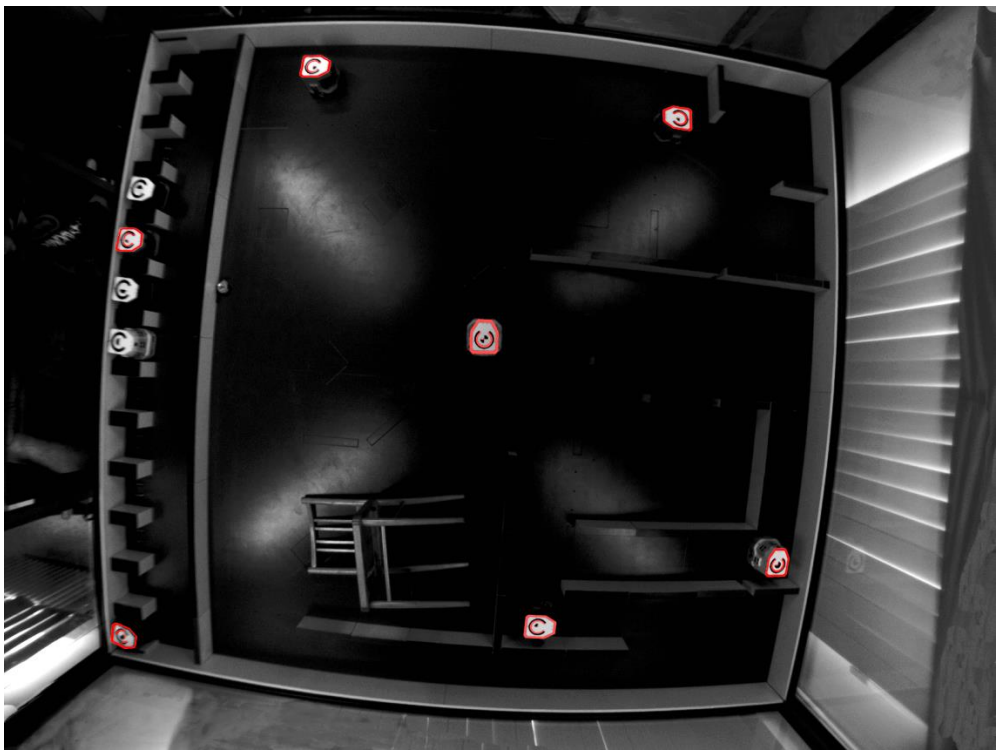
$$\mathbf{h} = \begin{pmatrix} \mathrm{h}^1 \\ \vdots \\ \mathrm{h}^9 \end{pmatrix}, \qquad H = \begin{pmatrix} \mathrm{h}^1 & \mathrm{h}^2 & \mathrm{h}^3 \\ \mathrm{h}^4 & \mathrm{h}^5 & \mathrm{h}^6 \\ \mathrm{h}^7 & \mathrm{h}^8 & \mathrm{h}^9 \end{pmatrix}. \tag{5.4}$$

Matice H je konečným výstupem mého programu.

6. Testování programu

V průběhu testování programu se ukázalo, že program není dostatečně robustní vzhledem k proměnnému osvětlení arény systému SyRoTek. Problém nastal především ve chvíli, kdy jediným zdrojem světla bylo umělé osvětlení nad hřištěm. V takové chvíli byl přechod mezi černou vodorovnou plochou a bílou svislou plochou obvodové stěny moc malý na to, aby mohl být detekován. Hlavními dvěma příčinami problému byla nedostatečná světlost „bílých“ stěn a svislá orientace bíle stěny. Za účelem nápravy tohoto problému jsem po obvodu hřiště nalepil bílou PVC pásku. Následkem toho se přechod posunul o cca. 15mm, ale také se zároveň přesunul na vodorovnou plochu obvodové stěny. Tato modifikace hřiště měla kýžený následek – hrany byly spolehlivě detekovány i v případě použití samostatného umělého osvětlení nad hřištěm.

Pro účely testování funkčnosti programu jsem rozmístil v aréně 7 robotů. Polohu těchto robotů jsem změřil svinovacím metrem. Následně jsem pořídil tři snímky plochy hřiště při různých vychýleních a natočeních kamery. Jeden z těchto tří snímků je vidět na obr. 13. Tomuto snímku odpovídá tab. 2, viz níže. Roboty, jejichž polohu určuji, jsou v obraze vyznačeny červenou barvou.



Obr. 13: Testování funkčnosti a přesnosti získané transformace.

Zbylé dva obrázky, na kterých bylo prováděno testování metody, jsou přiloženy na konci práce. Pro každý ze snímků byla programem vypočtena transformace mezi polohou v obraze a polohou v ploše hřiště. Porovnání poloh robotů získaných transformací a naměřených poloh je v tab. 1 – 3*.

	Souřadnice robotů získané transformací [mm]	Naměřené souřadnice robotů [mm]	Odchylka [mm]	
Robot č. 1	540,5	546	5,5	8,0
	1485,9	1480	5,9	
Robot č. 2	865,2	868	2,8	7,2
	284,7	278	6,7	
Robot č. 3	3031,1	3033	1,9	6,2
	957,9	952	5,9	
Robot č. 4	1870,5	1871	0,5	7,7
	1833,7	1826	7,7	
Robot č. 5	3167,8	3172	4,2	5,1
	2767,9	2765	2,9	
Robot č. 6	189,4	192	2,6	4,5
	3643,7	3640	3,7	
Robot č. 7	2261,2	2265	3,8	5,8
	3649,4	3645	4,4	

Tab. 1: Porovnání souřadnic získaných transformací a naměřených souřadnic pro obraz č. 1.

	Souřadnice robotů získané transformací [mm]	Naměřené souřadnice robotů [mm]	Odchylka [mm]	
Robot č. 1	539,0	546	7,0	7,1
	1481,2	1480	1,2	
Robot č. 2	864,0	868	4,0	4,1
	278,9	278	0,9	
Robot č. 3	3032,9	3033	0,1	3,9
	955,9	952	3,9	
Robot č. 4	1869,4	1871	1,6	6,1
	1831,9	1826	5,9	
Robot č. 5	3167,2	3172	4,8	4,8
	2764,9	2765	0,1	
Robot č. 6	188,6	192	3,4	4,4
	3637,2	3640	2,8	
Robot č. 7	2260,1	2265	4,9	6,8
	3640,4	3645	4,6	

Tab. 2: Porovnání souřadnic získaných transformací a naměřených souřadnic pro obraz č. 2.

* Pro tabulky 1-3 platí, že hodnoty v prvním řádku u každého ze sedmi robotů patří k x-ové souřadnici a hodnoty v druhém řádku patří k y-ové souřadnici. V posledním sloupci je hodnota získána kombinací zmíněných dvou hodnot ($o = \sqrt{o_x^2 + o_y^2}$, kde o je kombinovaná odchylka a o_x, o_y jsou odchylky jednotlivých souřadnic).

	Souřadnice robotů získané transformací [mm]	Naměřené souřadnice robotů [mm]	Odchylka [mm]	
Robot č. 1	539,8	546	6,2	7,1
	1483,6	1480	3,6	
Robot č. 2	865,3	868	2,7	2,7
	278,4	278	0,4	
Robot č. 3	3030,6	3033	2,4	3,7
	954,7	952	2,7	
Robot č. 4	1871,3	1871	0,3	7,1
	1833,1	1826	7,1	
Robot č. 5	3171,1	3172	0,9	3,0
	2767,9	2765	2,9	
Robot č. 6	186,1	192	5,9	6,2
	3642,0	3640	2,0	
Robot č. 7	2260,6	2265	4,4	5,3
	3648,0	3645	3,0	

Tab. 3: Porovnání souřadnic získaných transformací a naměřených souřadnic pro obraz č. 3.

	průměrná odchylka [mm]	min. odchylka [mm]	max. odchylka [mm]
Obraz č. 1	6,4	4,5	8,0
Obraz č. 2	5,3	3,9	7,1
Obraz č. 3	5,0	2,7	7,1

Tab. 4: Porovnání odchylek jednotlivých obrazů.

6.1. Diskuze výsledků

Z výše uvedených tabulek je vidět, že přesnost určení polohy je horší než u stávajícího řešení, které dokáže určit polohu robotu s přesností ± 3 mm [8, s. 8]. Na rozdíl od stávajícího statického výpočtu transformace se musí totiž můj program potýkat s proměnlivými podmínkami a robustně reagovat na změny ve scéně zachycené kamerou. Například při různé intenzitě osvětlení hřiště se mění profil detekované hrany z téměř ideální skokové hrany na širší lineárně rostoucí hranu. Následkem toho se může pozice, na které hranu detekuji, lišit o 1 pixel. V ploše hřiště odpovídá 1 pixel 3,5 mm, z čehož je jasné, že i „pouhý“ posun hrany o 1 pixel, výrazně zhorší přesnost celé transformace. Tyto skutečnosti jsou dány charakterem řešeného problému a nejsou pro mne jednoduše ovlivnitelné. Z těchto důvodu považuji dosažený výsledek za úspěšný. Především pak skutečnost, že přesnost určení transformace není závislá na vychýlení kamery a rotaci kamery o jednotky stupňů z původní pozice, jak je patrné z tab. 4.

7. Závěr

V předložené práci bylo zpracováno téma zabývající se zpracováním obrazu. Nejprve se práce zaměřovala na teorii tohoto tématu, konkrétně na digitalizaci obrazu, hledání hran v obraze a projektivní transformaci. Na teorii první části byla vystavěna následující praktická část, která se zabývala řešením úlohy na projektu SyRoTek. V praktické části byla podrobně popsána implementace řešené úlohy a na závěr byly uvedeny data získaná při testování programu.

Cílem práce bylo navrhnout metodu, která by umožnovala dynamické získávání transformace mezi obrazem z kamery a plochou hřiště uvnitř arény systému SyRoTek, resp. kalibraci této transformace. Tento cíl byl splněn a mnou implementovaná metoda je funkční. Jak již bylo uvedeno v předchozí kapitole, přesnost transformace, resp. její odchylka dosahuje v průměru přibližně 5,5 mm Možným způsobem pro dosáhnutí větší přesnosti, by bylo, podle mého názoru, hledání polohy obvodové hrany se snahou dosažení sub-pixelové přesnosti. Program by nejprve našel hranu s přesností ± 1 pixel a poté by určil z gradientního obrazu co nejpřesnější polohu hrany. Otázkou však je, zda by takové řešení bylo vůbec realizovatelné. Také by bylo nejspíš mnohem složitější, takže tato možnost může být využita jako námět pro další práci. Přesnost vytvořeného programu by i přesto měla být dostatečná na to, aby systém pracoval správně.

Mnou vytvořený program by měl do budoucna především zvýšit spolehlivost celého systému. V případě, že by bylo nasměrování kamery pozměněno, např. při neopatrné manipulaci uvnitř arény, dokáže program na tuto situaci zareagovat a nalézt novou transformaci, která bude znovu funkční. Tímto bude jednak eliminována nutnost zásahu do systému správcem, ale zároveň také systém poběží bez nutnosti odstávky. V tomto spatřuji hlavní přínos své práce. Dalším přínosem bylo získání nových znalostí a zkušeností s prací na větším, reálném problému.

Seznam literatury

1. HLAVÁČ, Václav a Miloš SEDLÁČEK. *Zpracování signálů a obrazů*. 1. vyd. Praha: Nakladatelství ČVUT, 2001, 220 s. ISBN 80-01-02114-9.

2. BOVIK, Al. *The Essential Guide to Image Processing*. 2. vyd. Burlington, USA: Elsevier, 2009. ISBN 978-0-12-374457-9.

3. SNYDER, Wesley E. a Hairong QI. *Machine Vision*. Cambridge, UK: Cambridge University Press, 2004. ISBN 0-521-83046-X.

4. RUSS, John C. *The image processing handbook*. 6th ed. Boca Raton: CRC Press, 2011, xviii, 867 p. ISBN 978-143-9840-450.

5. JAIN, Ramesh, Rangachar KASTURI a Brian G. SCHUNCK. *Machine vision*. Vyd. 1. Boston: McGraw-Hill, 1995, 549 s. ISBN 00-703-2018-7.

6. HARTLEY, Richard a Andrew ZISSERMAN. *Multiple View Geometry: in computer vision*. 2. vyd. Cambridge, UK: Cambridge University Press, 2003. ISBN 978-0-521-54051-3.

7. CANNY, John F. A Computational Approach to Edge Detection. In: *IEEE transactions on pattern analysis and machine intelligence*. Los Alamitos: IEEE Computer Society, 1986, s. 679-698. ISSN 0162-8828. DOI: PAMI-8(6).

8. CHUDOBA, Jan, Miroslav KULICH, Tomáš KRAJNÍK, Karel KOŠNAR a Libor PŘEUČIL. *A TECHNICAL SOLUTION OF A ROBOTIC E-LEARNING SYSTEM IN THE SYROTEK PROJECT*. Praha, 2011. Dostupné z: https://syrotek.felk.cvut.cz/data/files/csedu11.pdf. In Proc. of 3rd international conference on computer supported education. ČVUT FEL Katedra kybernetiky.

9. Hough transform. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-05-02]. Dostupné z:http://en.wikipedia.org/wiki/Hough_transform.

10. GANDER, Walter. *The Singular Value Decomposition*. EHT Zurich, 2008. Dostupné z:http://www.math.ethz.ch/education/bachelor/lectures/hs2012/other/linalg_INFK/svdneu.pdf. Přednáška. EHT Zurich.

11. GOLUB, Gene H. a C. REINSCH. Singular value decomposition and least squares solutions. In: *Numerische Mathematik*. Heidelberg: Springer, 1970, 403 - 420. ISSN 0029-599x.

12. *GNU: GSL - GNU Scientific Library* [online]. 1996, 27.4.2014 [cit. 2014-05-08]. Dostupné z: http://www.gnu.org/software/gsl/.

Seznam obrázků

A. AUTOR NEUVEDEN. *Newsplies* [online]. [cit. 12.5.2014]. Dostupný na WWW: newsplies.com/emma-watson-is-temporarily-removed-from-the-film-for-his-studies/emma-watson-pictures-6/.

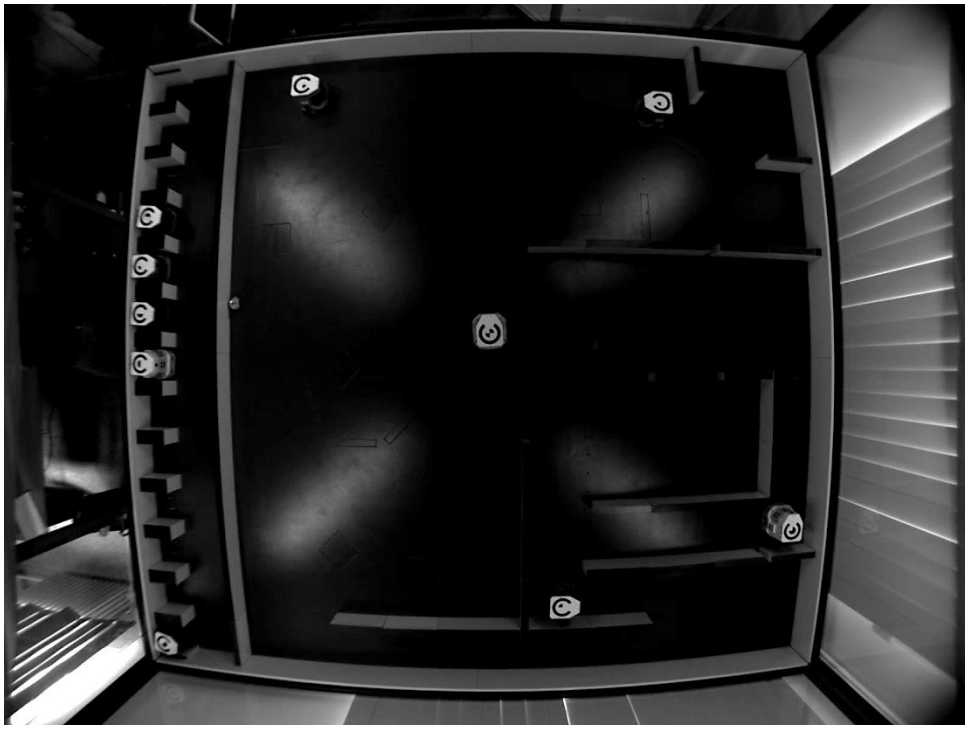
B. AUTOR NEUVEDEN. *Automobiles review* [online]. [cit. 12.5.2014]. Dostupný na WWW: http://www.automobilesreview.com/gallery/2013-dodge-viper-srt/2013-dodge-viper-srt-92-1.jpg.

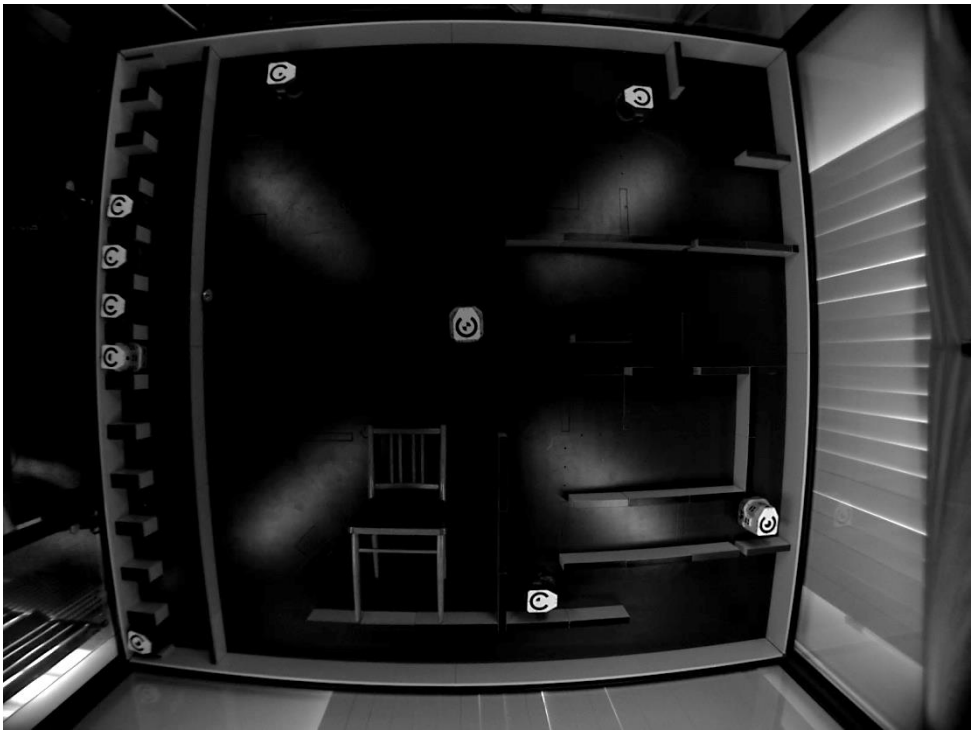
C. HARTLEY, Richard a Andrew ZISSERMAN. *Multiple View Geometry: in computer vision*. 2. vyd. Cambridge, UK: Cambridge University Press, 2003. ISBN 978-0-521-54051-3.

Přílohy

A. Obrazy z kamery využité při testování

Níže uvedené obrazy byly použity pro testování funkčnosti implementované metody (kapitola 6). Výsledky pro první z uvedených obrázků jsou uvedeny kapitole 6 v tab. 1 a výsledky pro druhý obrázek jsou uvedeny v tab. 3.





B. Zdrojový kód programu

Na přiloženém DVD/CD se kromě elektronické verze práce ve formátu PDF nachází také složka „Zdrojový kód programu“, která obsahuje následující soubory se zdrojovým kódem programu:

Brightness.cpp
Brightness.h
Constants.h
CornersExtracting.cpp
CornersExtracting.h
Edge.cpp
Edge.h
EdgeCreating.cpp
EdgeCreating.h
EdgeDetect.cpp
EdgeDetect.h
KC.mat
KK.mat
Pixel.cpp
Pixel.h
Transformation.cpp
Transformation.h
Unbarrel.cpp
Unbarrel.h