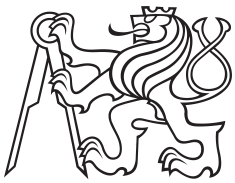
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Lokalizace hran v obraze za účelem přesného polohování

Anna Žigajkova

Obor: Kybernetika a robotika
Vedoucí: Ing. Pavel Krsek, Ph.D.
Praha, Srpen 2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Žigajkova** Jméno: **Anna** Osobní číslo: **474399**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra kybernetiky**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Lokalizace hran v obraze za účelem přesného polohování

Název bakalářské práce anglicky:

Edge Localization in the Image for Precise Positioning

Pokyny pro vypracování:

1. Seznamte se s algoritmy hranové detekce a automatického ostření.
2. Prostudujte možnosti realizace hranové detekce v NI Vision Bulider a navrhněte způsob začlenění vlastního kódu.
3. Navrhněte a implementujte algoritmus hranové detekce (upravte standardní algoritmus).
4. Experimentálně ověřte funkčnost a přesnost (kvalitu) navrženého algoritmu.
5. Porovnejte výsledky navrženého algoritmu se standardními algoritmy implementovanými v NI Vision Builder.

Seznam doporučené literatury:

[1] Milan Sonka, Vaclav Hlavac, and Roger Boyle. Image Processing, Analysis and Machine Vision. Thomson, 3rd edition, ISBN 978-0-495-08252, 2007.
[2] Hartley, Richard and Zisserman, Andrew. Multiple view geometry in computer vision. Cambridge University, 2nd edition, ISBN 0-521-54051-8, 2003.
[3] J. Canny. A computational approach to edge detection. IEEETransactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, pp. 679–698, Nov 1986.
[4] S. Kraus, Měření rozměrů CCD kamerou se submikronovou přesností. Diplomová práce, katedra řídicí techniky, FEL, ČVUT v Praze, 1999.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Pavel Krsek, Ph.D., robotické vnímání CIIRC

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **10.01.2020** Termín odevzdání bakalářské práce: **14.08.2020**

Platnost zadání bakalářské práce: **30.09.2021**

Ing. Pavel Krsek, Ph.D.	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studentky

Poděkování

Chtěla bych poděkovat vedoucímu práce Ing. Pavlu Krskovi Ph.D. za cenné rady a odborné vedení při vypracování této práce.

Prohlášení

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

V Praze, 13. srpna 2020

Abstrakt

Tato práce se zabývá lokalizací hran na snímcích při přesném polohování součástek ve výrobě integrovaných optických modulů. Byl implementován algoritmus na detekci hranových bodů a jejich aproximaci analytickou křivkou. V rámci testování algoritmu byly vyzkoušeny tři metody aproximace hranových bodů (metodou nejmenších čtverců, metodou vážených nejmenších čtverců, RANSAC). Navržený algoritmus byl připojen do aplikace NI Vision Builder a porovnán s dostupným hranovým detektorem. Ukázalo se, že algoritmus s metodou aproximace RANSAC dosahuje lepších výsledků na snímcích v této aplikaci než detektor implementovaný v programu NI Vision Builder.

Klíčová slova: hranová detekce, aproximace analytickou křivkou, automatické ostření, NI Vision Builder, LabVIEW, optické měření, přesné polohování

Abstract

This work proposes an edge detection algorithm for manufacturing of integrated optical modules.The algorithm detects edglets and approximates them with analytical curve. Three edglet approximation methods were explored: least-squares, weighted least-squares, RANSAC. The proposed algorithm was imported to NI Vision Builder, achieving better results than the built-in NI Vision Builder edge detection algorithm.

Keywords: edge detection, analytical curve approximation, autofocus, NI Vision Builder, LabVIEW, precise positioning, optical measurement

Title translation: Edge Localization in the Image for Precise Positioning

Obsah

Kapitola 1
Úvod

Hlavním cílem této práce je navrhnout a realizovat algoritmus detekce hran, který bude základem pro přesné polohování. Bylo realizováno začlenění vlastního detektoru hran do prostředí NI Vision Builder, který je používán pro specifikaci úlohy optického měření.

Tato práce vznikla v rámci projektu řešeného firmou EZconn technologies CZ s.r.o. ve spolupráci s ČVUT. Tato firma se zabývá výrobou mikrosystémů na zakázku, zaměřuje se primárně na výrobu polovodičových optických vysílačů a dalšího vybavení optických sítí. V rámci výše zmíněného projektu je vyvíjen systém řízení letovací buňky součástek při výrobě optoelektrických modulů.

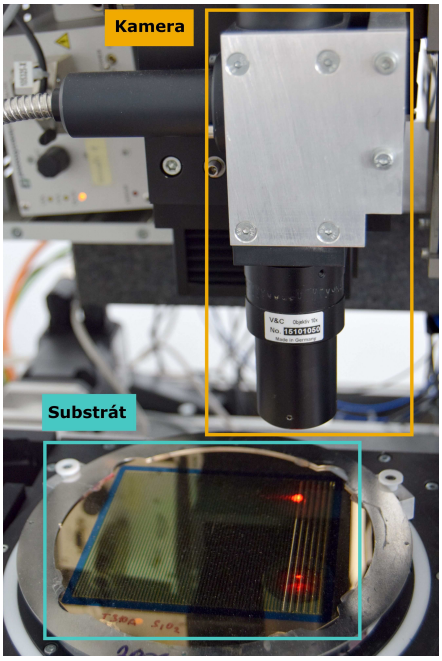
Buňka obsahuje dvě pracoviště. Na zdrojovém pracovišti jsou umístěny součástky v rastru na tenké bláně. Manipulátor pomocí duté jehly s podtlakem přenáší součástky na cílové pracoviště, viz obrázek 1.1. Na cílovém pracovišti jsou součástky přesně umístěny vůči obrazci na substrátu, který je podobný desce plošných spojů (obr. 1.2), a ostatním osázeným součástkám. Poté je součástka přitlačena na substrát a přiletována laserem. Pro správné umístění součástky je nutné znát její aktuální polohu a cílovou polohu. K tomu se používá měření polohy kamerou s telecentrickým objektivem. Pro přitlačení k substrátu (osa z) se používá tlakový senzor. Tato kombinace dvou metod měření je výhodná i z hlediska následného letování, kdy je potřeba přitisknout součástku na substrát předepsanou silou.

Přibližná poloha součástky je známa. Optické měření stanoví polohu s přesností požadovanou pro osazování (menší než 1 μm). Operátor volí obvykle dvě kolmé hrany podle kterých se určuje poloha součástky při osazování. Poloha se určuje vůči známému vzoru substrátu, který je tvořen kresbou z úseček. Z hlediska měření na snímku jsou detekovány úsečky tvořené hranami různých kovových vrstev na substrátu a hranami ostatních součástek. Podstatou detekce měření je proto detekce hran. Měření potřebné pro umístění součástky není možné provést v jednom snímku, jelikož hloubka ostrosti je nedostačující pro zobrazení součástky a substrátu zároveň (obrázek 1.3).

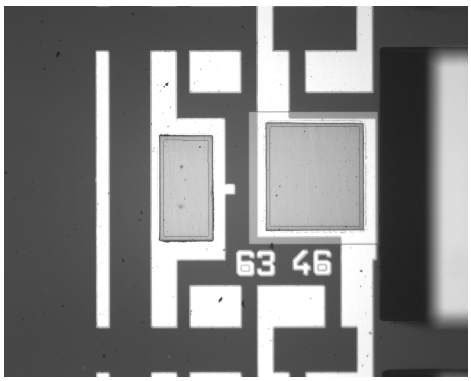
Doposud byla detekce hrany prováděna pomocí aplikace National Instruments Vision Builder for Automated Inspection (*NI Vision Builder, VBAI*), které se zaměřuje na zpracování dat robotického vidění a automatizovanou kontrolu výrobku. Tato aplikace poskytuje možnost hledání hrany ve vybrané oblasti s možností volby základních parametrů mezi které patří například směr barevného přechodu a zda je hledána první požadovaná hrana anebo

nejlepší hrana.

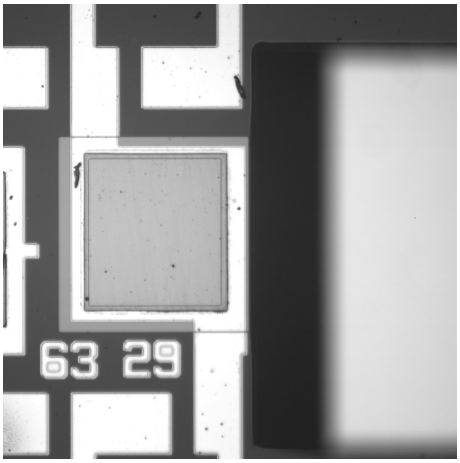
Na substrátu a součástkách se nacházejí známé vady s nimiž je nutno počítat. Například jednou ze součástek jejíž polohu je nutno znát je zrcátko, které je řezáno diamantovými kotoučky. Při tom může docházet ke štípání na řezaných hranách. Nejde o vadu výrobku, ale je možné s tím počítat při optickém měření. Na snímcích je možné pozorovat také vícenásobné hrany na substrátu, které vznikají v důsledků nepřesných dosednutí masek při nanášení kovových vrstev.



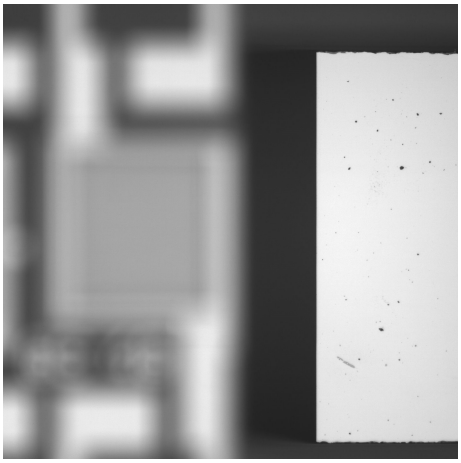
Obrázek 1.1: Cílové pracoviště letovací buňky.



Obrázek 1.2: Substrát pro optoelektronické systémy.



(a) : Kamera je zaostřena na vzor substrátu.



(b) : Kamera je zaostřena na zrcátko.

Obrázek 1.3: Substrát s umístěným zrcátkem.

Kapitola 2
Hranová detekce

V rámci této práce se budou zpracovávat sekvence jednobarevných snímků pořízené monochromatickou kamerou s telecentrickým objektivem. Jednotlivé snímky představují dvojrozměrné matice s počtem řádků a sloupců odpovídajícím rozlišení pořízeného snímku. Prvky této matice zaznamenávají příslušné hodnoty obrazové funkce, která v případě černobílého snímku odpovídá jasové funkci $I(u, v)$, kde u, v jsou souřadnice na snímku.

V digitálním zpracování obrazu je zvykem označovat tyto prvky jako *obrazové body*, častěji *pixely*. Obrazová funkce popisuje intenzitu odraženého světla, které vstoupilo do objektivu kamery a dopadlo na snímací čip. Hrany na snímcích vznikají v důsledku existujících změn odrazivosti na scéně. Hrana ve scéně se promítá na čip kamery, kde dochází k diskretizaci hrany. Pixely příslušné k hraně na snímku se označují jako *hranové body*.

Dá se usuzovat, že v blízkém okolí hranových bodů dochází k velké změně obrazové funkce. Tuto změnu můžeme vyšetřovat na základě gradientu. Je možné definovat velikost gradientu jasové funkce (rovnice (2.1)) a směr gradientu v daném bodě (u, v) (rovnice (2.2)) dle [1, str. 78].

$$|\nabla I(u,v)| = \sqrt{\left(\frac{\partial I(u,v)}{\partial u}\right)^2 + \left(\frac{\partial I(u,v)}{\partial v}\right)^2} \tag{2.1}$$

$$\varphi = \arctan\left(\frac{\frac{\partial I(u,v)}{\partial u}}{\frac{\partial I(u,v)}{\partial v}}\right) \tag{2.2}$$

Řež jasovou funkcí ve směru gradientu (kolmo k hraně) je zobrazen na obrázku 2.1.

Digitální obraz reprezentuje diskrétní funkci jasu, z toho důvodu jsou následující algoritmy popsány v diskrétní doméně. K výpočtu aproximace derivací a filtraci vstupního snímku se využívá konvoluce (dále značeno operátorem $\circledast$). Konvoluci vstupního snímku $I(u, v)$ s konvolučním jádrem h definujme jako

$$I(u,v) \circledast h = \sum_i \sum_j I(u-i, v-j) \cdot h(i,j), \tag{2.3}$$

kde i, j jsou parametry, pro které jsou definovány hodnoty funkcí $h(i, j)$ a $I(u - i, v - j)$.

2.1 Odstranění šumu a filtrace

Při snímání reálnou kamerou na snímku je zaznamenán i šum s poissonovým rozdělením s parametrem $\lambda > 0$ pro všechny hodnoty $x \in \mathbf{N}$ náhodné veličiny X

$$P(X = x) = \frac{\lambda^x}{x!} e^{-\lambda}, \tag{2.4}$$

který je při detekci hran výpočtem gradientu snímku zvýrazněn spolu s hranami. Tento jev nastává v důsledku srovnatelně vysokých frekvencí jak nežádoucího šumu, tak hledaných hran. Tomuto jevu se dá předcházet filtrací vstupního snímku. Nejjednodušší možnou filtrací je *průměrování*, které se dá provádět ve dvou možných variantách. Pokud existuje více snímku ve stejné poloze, je možné provést průměrování přes všechny snímky a získat jeden výstupní obrázek. Pokud je k dispozici pouze jeden snímek, je možné jej filtrovat konvolucí s konvoluční maskou, jejíž velikost je závislá na charakteru šumu. Masku s rozměrem 3×3 pixelů můžeme definovat:

$$h = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \tag{2.5}$$

viz [1, str. 69-71]. Dalším používaným filtrem je *Gaussův filtr* (rovnice (2.6)) s parametrem směrodatné odchylky σ příslušného normálního rozdělení [1, str. 84].

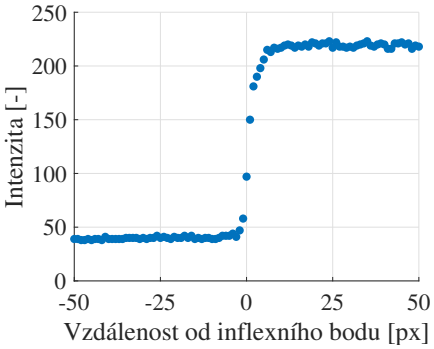
$$G(u, v) = e^{-\frac{u^2+v^2}{2\sigma^2}} \tag{2.6}$$

Tento filtr se dá aproximovat konvolučními maskami v závislosti na požadované váze bodů v okolí. Příklady masek velikosti 3×3 pixely:

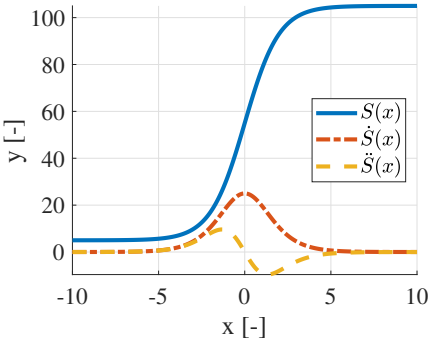
$$h_G = \frac{1}{10} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad h_G = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}. \tag{2.7}$$

2.2 Detektory hranových bodů

Při zpracování digitálních snímku se parciální derivace jasové funkce nahrazují diferencemi dle [1, str. 78]. V počítačovém vidění, pro omezení výpočetní náročnosti, bývají využívány aproximace diference ve čtyřech nebo v osmi směrech. Běžným postupem při zpracování obrazu je ovšem použití některého z *hranových detektorů*. Hranové detektory jsou založeny na detekci výrazné změny intenzity jasové funkce, která je přítomna v okolí hranových bodů. Jedná se o takzvané hranové operátory, které jsou ze své podstaty konvolučními filtry s jádry, která reprezentují numerický výpočet derivace v daném bodě a směru. Při konvoluci jádra se vstupním snímkem jsou získány hledané složky diference. Algoritmy pro detekci hran v počítačovém vidění je možné rozdělit na tři skupiny:



Obrázek 2.1: Řez hrany ve směru největšího spádu jasové funkce.



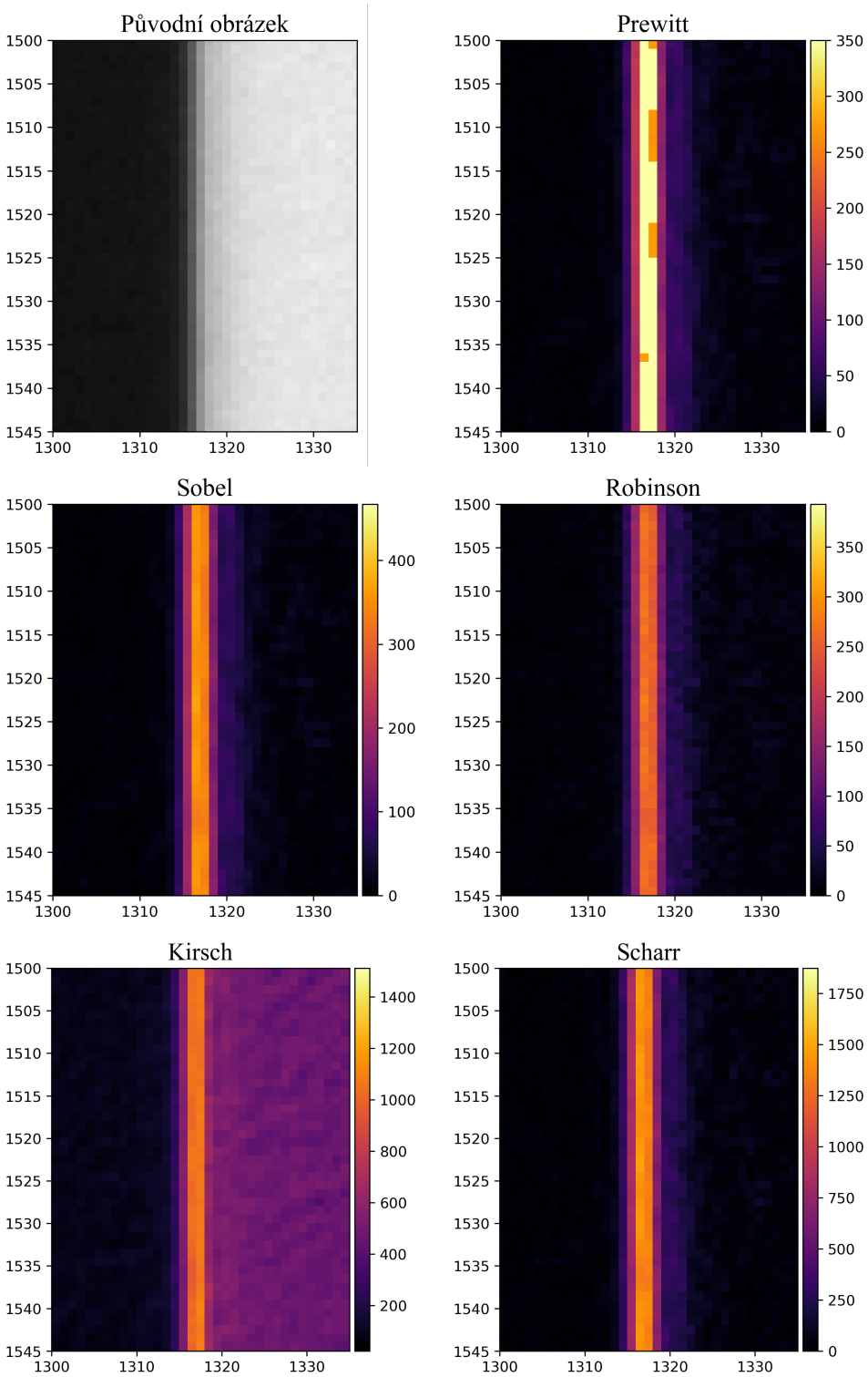
Obrázek 2.2: Ukázka křivky $S(x) = \frac{100}{1+e^{-x}} + 5$ pro aproximaci hranových bodů vyskytujících se na substrátu a její první $\dot{S}(x)$ a druhá $\ddot{S}(x)$ derivace.

- Detektory prvního řádu - hledání maxima první derivace.
- Detektory druhého řádu - hledání průchodů nulou derivace druhého řádu.
- Parametrické modely - popis hrany pomocí parametrických modelů.

2.2.1 Hranové detektory prvního řádu

Detektory pracující na principu hledání lokálních extrémů v obraze vychází z předpokladu, že v okolí hrany dochází k největší změně intenzity jasové funkce a na okolních homogenních plochách je změna jasové funkce minimální, tedy její gradient je blízký nule (obr. 2.2). Tyto detektory pomocí konvolucí s hranovými operátory aproximují hodnotu první diference v obraze a následně hledají lokální extrémy. Výhodou využití hranových detektorů prvního řádu je jejich schopnost rozlišit směr gradientu, která umožňuje pro specializované případy předem předtřídit hledané hranové body. Mezi nejčastěji používané operátory dle [1] patří *Prewittův operátor*, *Sobelův operátor*, *Kirschův operátor* a *Scharrův operátor*. Scharrův operátor aproximuje gradient ve dvou směrech. Ostatní operátory aproximují derivaci v osmi směrech. S tím, že konvoluční maska, která v daném bodě poskytne maximální odezvu v absolutní hodnotě určuje směr gradientu v příslušném bodě. Dále jsou v tabulce 2.1 uvedeny příklady tří konvolučních masek všech operátorů s tím, že ostatní jsou od nich odvozené rotací.

Na obrázku 2.3 jsou příklady odezvy operátorů na hraně snímku. Můžeme vidět, že v tomto případě nejjistěji detekoval hranu Scharrův operátor. Na rozdíl od Prewittova, Sobelova a Robinsonova operátoru dokázal lépe detekovat rozdíl mezi hranou a homogenním okolím. V porovnání s Kirschovým operátorem se ukazuje jako výhodnější, protože je méně závislý na jasu původního snímku.



Obrázek 2.3: Porovnání odezvy operátorů hranových detektorů prvního řádu. Barevné stupnice příslušné jednotlivým grafům znázorňují maximální odezvy jednotlivých pixelů pro tento typ operátoru. Čím je větší odezva pixelu, tím spíše se bude na jeho pozici nacházet hranový bod.

Operátor	$\mathbf{h_1}$	$\mathbf{h_2}$	$\mathbf{h_3}$
Prewitt	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$
Sobel	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
Robinson	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
Kirsch	$\begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix}$	$\begin{bmatrix} 3 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & -5 & 3 \end{bmatrix}$	$\begin{bmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{bmatrix}$

Operátor	$\mathbf{h_x}$	$\mathbf{h_y}$
Scharr	$\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix}$	$\begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix}$

Tabulka 2.1: Hranové operátory aproximující první derivaci. Scharrův operátor je převzat z [4, str. 120], ostatní z [1, str. 80-81].

2.2.2 Hranové detektory druhého řádu

Tyto hranové detektory hledají hranové body pomocí detekce nulovosti druhé derivace (obr. 2.2). Jelikož se ovšem tento bod může nacházet mimo zobrazovanou mřížku je nutné vycházet z předpokladu, že veškeré body jejichž norma derivace je nižší než určitá mez mohou být hledanými inflexními body. Tyto detektory jsou citlivější na šum a je tedy nutné je kombinovat s filtry. Nejběžnějším detektorem z této kategorie je *Marr-Hilderthův* hranový detektor [1, str. 83]. Tento detektor filtruje vstupní snímek pomocí filtru, který splňuje tyto podmínky

- Filtr má být hladký a má být frekvenčně pásmovou propustí, která potlačí vyšší frekvence odpovídající šumu.
- Filtr má reagovat pouze na ty body, které se nachází v blízkém okolí hrany.

Nejčastějším příkladem filtru s těmito vlastnostmi je výše zmíněný Gaussův filtr (2.6). Při aplikaci Gaussova filtru je dán důraz na pixely, které se na-

chází blíže středu filtračního okna. Je možné předpokládat, že vliv pixelů vzdálenějších o více jak 3σ je zanedbatelný.

V dalším kroku je aplikován *laplaceův operátor* (rovnice (2.8)).

$$\Delta = \nabla^2 = \frac{\partial^2 I(u,v)}{\partial u^2} + \frac{\partial^2 I(u,v)}{\partial v^2} \tag{2.8}$$

Obvykle se pro výpočet laplaceova operátoru používá konvoluční maska. (Příkladem konvolučních masek je (2.9))

$$h(u,v) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \; h(u,v) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{2.9}$$

Tyto konvoluční masky jsou invariantní vůči orientaci a udávají pouze sílu hranového bodu v daném místě a nikoli směr hrany. Spojením Gaussova filtru a Laplaceova operátoru při zpracování obrazu se používá operátor *Laplacián Gaussiánu* (*LoG*). Jelikož obě operace jsou lineární, je možné zaměnit pořadí derivace a konvoluce, jak je uvedeno ve vztahu (2.10).

$$\Delta \left[h_G(u,v) \circledast I(u,v) \right] = \left[\Delta h_G(u,v) \right] \circledast I(u,v) \tag{2.10}$$

Tyto kroky je možné spojit do jednoho (rovnice (2.11)) a vyjádřit je pomocí jedné konvoluční masky, podrobněji je tento postup popsán v [1, str. 84-85]. Nevýhodou použití tohoto operátoru je jeho snaha vyhladit ostré tvary, což nese za následek chybnou detekci rohů.

$$LoG(u,v) = c \left(\frac{u^2+v^2}{\sigma^2} - 1 \right) e^{-\frac{u^2+v^2}{2\sigma^2}}, \tag{2.11}$$

kde c je koeficient normalizace.

2.2.3 Aproximace hrany parametrickými modely

Pro přesnější určení polohy hranových bodů lze aproximovat jasové funkce parametrickými modely.

Aproximace S-křivkou

Hrany lze popsat různými parametrickými modely v závislosti na tělesech, která tvoří. Tato metoda byla popsána v disertační práci Svatopluka Krause a poskytuje subpixelovou přesnost [3, str. 72-80].

Pro lokalizaci hran je výhodné tvořit řezy jasové funkce ve směru gradientu. A následně hodnoty tohoto řezu prokládat S-funkcí $S(x)$ (rovnice (2.12)). Můžeme předpokládat, že hledané hranové body odpovídají inflexním bodům křivky, neboť v těchto bodech změna intenzity jasové funkce dosahuje svého maxima. A proto lze aproximací hranových bodů touto křivkou zpřesnit jejich polohu nalezením inflexního bodu.

$$S(x) = \frac{L}{1 + e^{-k(x-x_0)}} + b \tag{2.12}$$

Parametr L určuje škálování v ose y, která na snímku odpovídá intenzitě jasové funkce $I(u, v)$. Parametr k škáluje křivku v ose x, která odpovídá posunu na snímku ve směru gradientu, b určuje posun křivky po ose y a x_0 je hledaný inflexní bod.

Fazetový model

Postup na aproximaci hran parametrickým modelem pro složité tvary publikoval Robert M. Haralick [5]. Jim navržený postup v sobě zahrnuje rozdělení obrazu na jednotlivé oblasti podobných barev a následné určení jejich hranic, které představují hledané hrany. Pro zjednodušení výpočtu je dále definováno, že nejmenší oblast bude složena z $K \times K,\ K \in \mathbb{N}$ pixelů. Zaveďme transformaci souřadnic, kde pixel s polohou (u, v) je počátkem souřadnicového systému (r, c)

$$\begin{aligned}(u, v) &\to (r, c) = (0, 0),\\ (u-1, v) &\to (-1, 0),\\ (u, v-1) &\to (0, -1)...\end{aligned} \tag{2.13}$$

Hodnota ve stupních šedi je pak dána

$$I(u, v) = J(r, c) + n(r, c) = ar + bc + g + n(r, c), \tag{2.14}$$

kde $n(r, c)$ představuje šum, $J(r, c)$ je reálná hodnota pixelu nezatížená šumem a $a,\ b,\ g$ jsou koeficienty přímky, která je rovnoběžná se směrem gradientu v tomto pixelu.

Pomocí metody nejmenších čtverců se na čtverci $< -L, L > \times < -L, L >$, $L \in \mathbb{N}$ minimalizuje funkce

$$f(a,\ b,\ c) = \sum_{u=-L}^{L} \sum_{v=-L}^{L} \left(ar + bc + g - J(r, c)\right)^2. \tag{2.15}$$

Z čehož pro jednotlivé koeficienty $a,\ b,\ c$ plyne

$$a = \frac{3}{L(L+1)(2L+1)^2} \sum_{r=-L}^{L} r \cdot \sum_{c=-L}^{L} J(r, c), \tag{2.16}$$

$$b = \frac{3}{L(L+1)(2L+1)^2} \sum_{c=-L}^{L} c \cdot \sum_{r=-L}^{L} J(r, c), \tag{2.17}$$

$$g = \frac{3}{(2L+1)^2} \sum_{r=-L}^{L} \sum_{c=-L}^{L} J(r, c). \tag{2.18}$$

Aproximovaná hodnota intenzity jednotlivých pixelů je tedy rovna

$$H(r, c) = ar + bc + g. \tag{2.19}$$

Z bloku o $K \times K,\ K \in \mathbb{N}$ prvcích s počátkem souřadnicového systému ve zkoumaném pixelu se vybere hodnota s minimální chybou

$$e^2(u, v) = \sum_{r=-L}^{L} \sum_{c=-L}^{L} \left[H(r,\ c) - J(r,\ c)\right]^2. \tag{2.20}$$

Oblasti jsou tvořeny pixely se stejnými koeficienty a, b, g. Na pomezí dvou oblastí se nachází hledaná hrana.

2.2.4 Cannyho detektor

Nejčastěji používaným hranovým detektorem v současných aplikacích je detektor popsaný Johnem Cannym v roce 1986 ve článku *A Computational Approach to Edge Detection* [6]. Tento algoritmus vychází z následujících požadavků:

- Požadavek na malou chybu. Významné hrany nesmí být přehlíženy a nesmí být detekovány chybné hranové body.
- Detektor by měl přesně lokalizovat střed hrany. Odchylka detekovaného hranového bodu od skutečného středového bodu hrany by měla být minimalizována.
- Žádná hrana v obraze nebude generovat vícenásobné odezvy.

V souladu s výše uvedenými požadavky byl Johnem Cannym navržen následující postup.

1. Na vstupní snímek je aplikován Gaussův filtr na minimalizaci šumu.
2. Jsou nalezeny hranové body.
3. Je aplikován algoritmus na ztenčení detekované hrany (*non-maximum suppression*). Pro každý bod je zhodnoceno jeho okolí, zda se jedná o pixel s nejvyšší hodnotou gradientu ve směru největšího spádu jasové funkce.
4. Na data je aplikováno prahování, pro stanovení potenciálních hranových bodů.
5. Hranové body jsou dohledány s použitím hystereze. Vychází se z potenciálních hranových bodů nalezených ve čtvrtém kroku. Tento postup potlačí méně výrazné hranové body, které nejsou spojeny s více výraznými hranovými body.

Výsledná mapa hranových bodů vzniká analýzou výstupů tohoto algoritmu pro různé hodnoty standardní směrodatné odchylky (velikosti) použité v Gaussově filtru.

2.3 Hledání hrany

Po identifikaci hranových bodů na snímků je možné hranu popsat jako analytickou křivku procházející těmito body. Nalezené body jsou proto aproximovány vhodnou analytickou křivkou. Na základě známých vlastností objektů na snímku je možné volit vhodné křivky pro aproximaci detekovaných bodů. Jak

je vidět na obr. 1.3a, v aplikacích zkoumaných touto prací je vhodné hranu aproximovat přímkou

$$p(u) = v = au + b,$$

kde a, b jsou parametry přímky.

Nejjednodušším aplikovatelným postupem je aproximace pomocí *metody nejmenších čtverců*, která minimalizuje sumu čtverců reziduí r_i (rovnice (2.21)). Reziduum můžeme definovat jako rozdíl hodnoty naměřené a aproximované. Tato metoda ovšem neposkytuje možnost vyřadit odlehlé hranové body, které byly detekovány na snímku chybně.

$$||r||^2 = \sum_i r_i^2, \tag{2.21}$$

kde r_i je hodnota rezidua $i-$*tého* bodu.

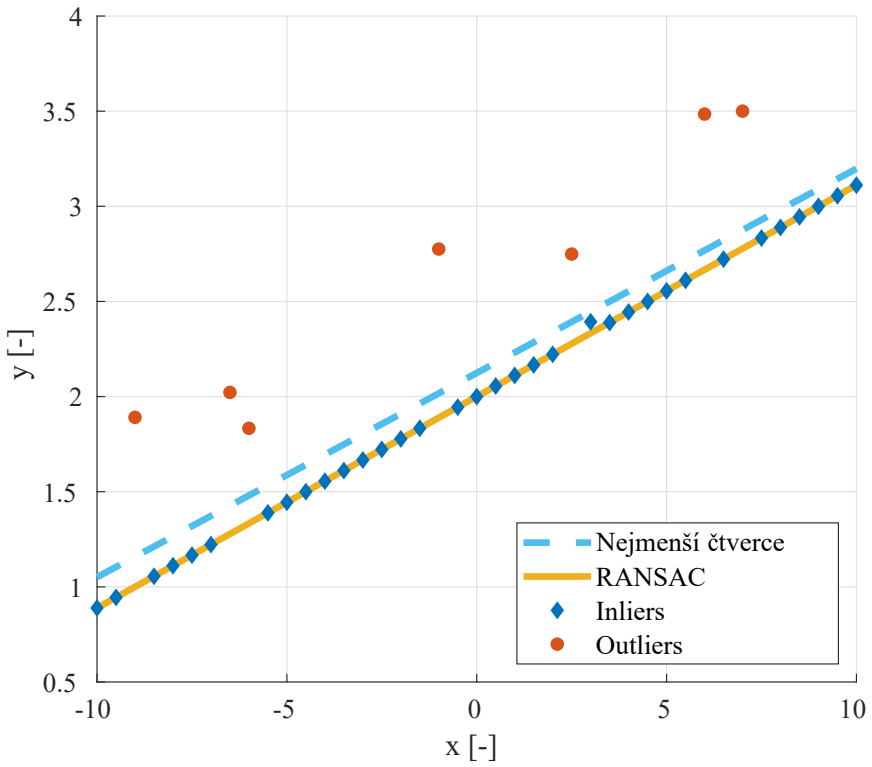
Pro snížení citlivosti metody na chybně detekované body je možné použít *metodu vážených nejmenších čtverců*. Bodům, jejichž polohou jsme si jisti, můžeme přiřadit vyšší váhu w_i. V tomto případě minimalizujeme výraz dle rovnice (2.22).

$$||r||^2 = \sum_i r_i^2 w_i, \tag{2.22}$$

kde w_i je váha $i-$*tého* bodu.

V případě měření zatíženého velkým počtem odlehlých bodů je výhodnější použít metodu *Random Sample Consensus* (*RANSAC*) popsanou v [2]. Tato iterativní metoda je schopna zanedbat vliv odlehlých bodů (*outliers*) na výslednou aproximaci. Při prokládání je použita jen podmnožina naměřených bodů bez odlehlých hodnot (*inliers*). Její algoritmus se dá shrnout do několika kroků. Nejdříve je vybrána náhodná množina bodů, které označíme za možné validní body. Tato množina bodů je proložena požadovanou křivkou. Poté ze všech měřených bodů je spočítána přesnost aproximace tímto modelem. Body, které dle zvolené ztrátové funkce odpovídají modelu, jsou přidány do podmnožiny potenciálních *inliers*. Spočítaný model je považován za dobrý, pokud dostatečný počet naměřených hodnot spadá do podmnožiny *inliers*. Tyto kroky jsou opakovány. Po stanoveném počtu opakování je vybrán nejlepší model ze všech pokusů.

Porovnání metody nejmenších čtverců s metodou RANSAC pro měření zatížené chybou, jakou lze pozorovat ve snímcích při měření, je na obrázku 2.4. Na náhodný počet bodů byla aplikována jednostranná chyba, která simuluje úlomky na hranách zrcátka, které jsou vidět například na obrázku A.1. Tato chyba byla modelována pseudonáhodným generátorem rovnoměrného rozdělení.



Obrázek 2.4: Porovnání proložení bodů přímky $y = \frac{1}{9}x + 2$ zatížené pseudonáhodně generovaným šumem s rovnoměrným rozdělením. Z grafu je vidět, že metoda RANSAC rozeznává dva typy bodů, tzv. *inliers*, které jsou relevantní pro prokládání hodnot přímkou, a tzv. *outliers*, což jsou body představující hodnoty zatížené šumem, které nejsou brány v potaz při prokládání hodnot. Metoda nejmenších čtverců proložila tento soubor bodů přímkou $y = 0,11x + 2,14$. Metoda RANSAC proložila inliers hodnoty přímkou $y = 0,11x + 2,00$.

Kapitola 3

Automatické ostření

Pro přesné optické měření je důležité zaostření obrazu. Snímaný objekt se musí nacházet v blízkosti roviny zaostření kamery. To je oblast, která se na snímku zobrazí ostře. Zaostřovací vzdálenost je vzdálenost roviny zaostření od kamery (ohniska). V případě aplikace popsané v této bakalářské práci, kdy je snímána oblast velikosti $1,5\ mm \times 1,2\ mm$ telecentrickým objektivem, je hloubka ostrosti malá. Použitý objektiv má pevnou ostřicí vzdálenost. Ostření obrazu se proto provádí změnou polohy kamery vůči snímané scéně. Ostří se podle uživatelem zvolené části obrazu. V této části by měla být alespoň jedna výrazná hrana, podle které je rozhodnuto o ostrosti obrazu.

Pro realizaci experimentů i samotné přesné polohování je nezbytné mít k dispozici zaostřený obraz. Opakované manuální ostření je časově náročně. Z tohoto důvodu současné kamery a systémy poskytují možnosti automatického ostření.

Moderní systémy automatického ostření se skládají ze senzoru, řídícího systému a pohonu, pomocí kterého se zaostří na vybraný objekt ve scéně. Dle postupu pro určení zaostření se systémy automatického ostření dají rozdělovat do dvou kategorií na systémy s aktivním ostřením a systémy s pasivním ostřením. V průmyslových a robotických aplikacích se setkáváme i s možností hybridního ostření, které kombinuje oba zmíněné systémy. Běžně se takté používá kombinace měření kontrastu a fázového posunu.

Patenty na první ostřící systémy byly publikovány už ve 30. letech minulého století, ovšem funkční modely začaly vznikat až v 70. letech, kdy americká firma Honeywell podala patent na aktivní ostřící systém založený na měření vzdáleností triangulací. S rozšířením digitálních fotoaparátů se ovšem od aktivních systému postupně ustupovalo. Prosadily se pasivní systémy, které jsou z hlediska výroby levnější, jelikož nevyžadují žádné dodatečné optoelektrické součástky. Mezi první příklad této technologie patří měření skrze objektiv (*Through the Camera Lens*). Jedná se o ostření založené na detekci fázového posunu. První kamera tohoto typu byla vyvinuta v druhé polovině 70. let. Tuto metodu dále následovala detekce kontrastu pomocí polohově citlivého (*CCD*) senzoru, která byla navržena již dříve, ale kamera s tímto postupem ostření byla vyrobena až v 80. letech. Při focení pohybujících se objektů se prosadili algoritmy takzvaného prediktivního autofokusu. Příkladem tohoto algoritmu je funkce *Continuous-servo AutoFocus (AF-C)*, která měří rychlost pohybu snímaného objektu a z ní se snaží odhadnout polohu objektu na snímku ve chvíli zmáčknutí spouště. [7]

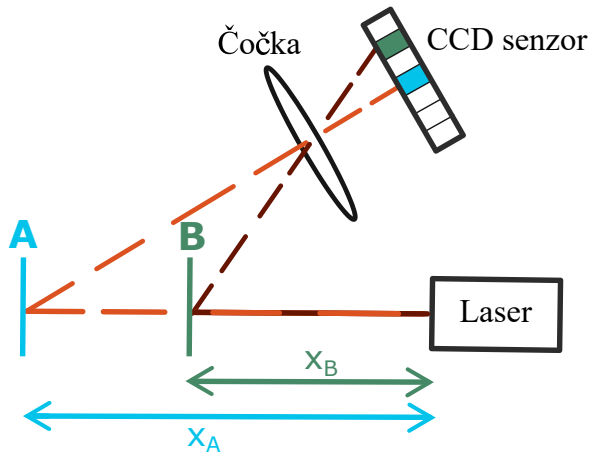
3.1 Aktivní ostření

Systémy s aktivním ostřením jsou opatřeny zdroji záření. Pomocí odrazu vyslaného paprsku od snímaného objektu se systém pokouší zjistit vzdálenost od objektu. Na scénu vysílá vybraný druh záření, jako ultrazvukový signál, infračervený nebo laserový paprsek. Odraz od objektů scény se snímá vhodným senzorem ve fotoaparátu a jeho analýzou se určuje vzdálenost.

U měření ultrazvukem je použita metoda měření doby letu (anglicky „*Time of Flight*“). Tato metoda v přímé variantě měří vzdálenost mezi kamerou a objektem ostření pomocí výpočtu vzdálenosti z doby uplynulé mezi vysláním signálu a návratem odraženého signálu do senzoru. Pro měření na kratší vzdálenosti je možné využít kombinace s metodou frekvenční modulace. Vzdálenost je v takovém případě určena z fázového posunu vysílaného a přijímaného odraženého signálu.

Infračervené měření využívá většinou principu laserových snímačů vzdálenosti s triangulačním měřením. Snímač vyhodnocuje vzdálenost objektu dle polohy jeho obrazu promítnutého na polohově citlivý senzor. Přesnost měření téměř nezávisí na stupni odrazivosti povrchu sledovaného objektu, pokud objekt nemá lesklý povrch. Tyto snímače jsou oproti běžně používaným reflexním snímačům odolnější vůči okolnímu optickému záření. Konstrukčně je před senzor umístěna čočka s funkcí optického filtru propouštějící pouze infračervené záření. Nákres měření je zobrazen na obrázku 3.1.

Výhodou aktivního ostření je možnost použití i při zhoršených světelných podmínkách, jelikož měření je založeno na vyslaném signálu. Nevýhodou těchto metod je malý dosah vysílaných paprsků, možnost nechtěného zakrytí externího čidla na těle přístroje nebo nefunkčnost při focení přes překážku, jako je sklo nebo pletivo.



Obrázek 3.1: Schéma měření vzdáleností triangulační metodou. Mějme dva objekty A a B, paprsky od nich odražené procházejí čočkou a dopadají na různá místa na polohově citlivý senzor (CCD senzor).

3.2 Pasivní ostření

Rozostření se na snímcích projevuje jako dolnofrekvenční propust. Při stejné scéně zaostřený snímek obsahuje více vyšších frekvencí, než snímek rozostřený. V důsledku menšího počtu vyšších frekvencí se hrany na snímku stávají méně ostré a malé objekty ztrácí svou intenzitu (šednou). Pasivní ostření se často používá u digitálních kamer. Ostřicí algoritmus zkoumá rozostření snímaného objektu z jednoho nebo více snímků a rozhoduje o směru doostření. Algoritmus se obecně skládá ze tří kroků. V prvním je vybrána zaostřovací oblasti. Následně je vyhodnocena ostrost vybrané oblastí. Na základě této informace je vypočtena nová poloha ostřicích prvků.

Pasivní ostření lze rozdělit na dvě hlavní skupiny. Na detekci fázového posunu, kdy je do soustavy přidána další optické součástka, a na detekci kontrastu vybrané oblasti.

3.2.1 Detekce fázového posunu

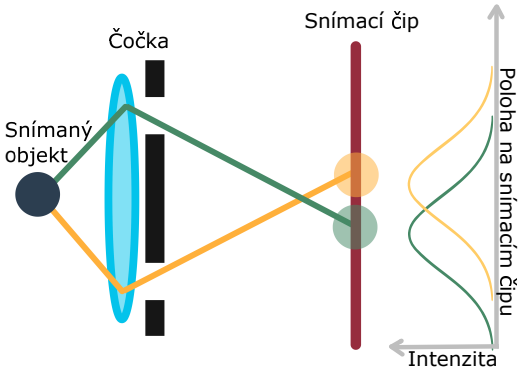
Tento postup je založen na pozorování objektu ze dvou míst (pravá a levá strana objektivu). Ostření pomocí detekce fázového posunu vyžaduje přidání optického prvku, který funguje jako dělící čočka. Tato čočka rozdělí vstupní paprsky a odkloní je na CCD senzor. Jednotlivé senzory vidí jen světlo přicházející z příslušné strany objektivu. Porovnáním údajů z nich je zjištěno, zda je snímaná scéna zaostřena. Pokud není zaostřeno, spočítá se směr posunu ostřicích prvků. Různé konfigurace kamery jsou zobrazeny na obrázku 3.2. Výhodou tohoto postupu je potřeba malého počtu kroků pro zaostření.

3.2.2 Detekce kontrastu

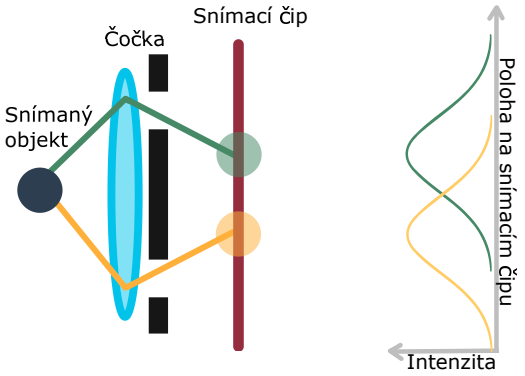
Tato metoda se zakládá na skutečnosti, že při maximálním kontrastu je obraz ostrý. Při detekci kontrastu je možné zkoumat míru rozostření na základě zaostřovacích funkcí. Mezi základní požadavky na zaostřovací funkce patří požadavek na existenci jediného globální extrému, kterého bude dosaženo při správném zaostření obrazu. Musí být zajištěna opakovatelnost zaostření a meze zaostřovacích vzdáleností by měly být co největší. Algoritmy by měly být aplikovatelné na různé scény a neměly by být závislé na vnějších vlivech. Měly by být kompatibilní s video signály a jednoduše implementovatelné. [8, str. 81]

Dle [8] lze rozdělit zkoumané ostřící funkce do tří kategorií na základě změn ve vyšších frekvencích a změn nastávajících na hranách snímaných objektů. Tyto funkce jsou založené na

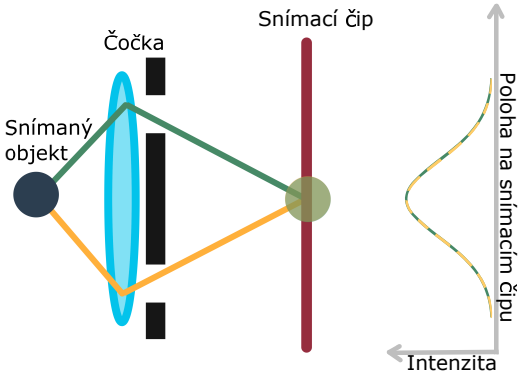
1. Derivaci jasové funkce $I(u, v)$.
2. Přímém měření jasové funkce $I(u, v)$.
3. Výpočtu odchylky od střední hodnoty jasové funkce $I(u, v)$.



(a) : Soustava je zaostřena za polohu snímaného objektu.



(b) : Soustava je zaostřena před polohu snímaného objektu



(c) : Soustava je zaostřena na snímaný objekt.

Obrázek 3.2: Příklady možných konfigurací optické soustavy při měření fázového posunu.

Funkce založené na derivaci jasové funkce

První typ ostřicí funkce $F^1_{n,m,\Theta}$ dle [8] popisuje změny vyšších frekvencí snímku jako součet hodnot gradientu snímku (rovnice (3.1)).

$$F^1_{n,m,\Theta} = \iint_{snímek} \mathbf{E}\Big(\left\| \frac{\partial^n I(u,v)}{\partial u^n} \right\| - \Theta \Big)^m \mathrm{d}u \; \mathrm{d}v, \tag{3.1}$$

kde Θ je libovolný práh a platí

$$\mathbf{E}(z) = \begin{cases} z & \text{pokud } z \geq 0, \\ 0 & \text{jinak.} \end{cases} \tag{3.2}$$

Výpočet velikosti gradientu Tento algoritmus používá parametry ostřicí funkce $n = 1$, $m = 1$, $\Theta = 0$. Metoda funguje dobře pro ostření objektů menších než velikost rozptylové funkce popisující tvar, do nějž se v zobrazovacím systému promítne bodový zdroj světla, protože výšky lokálních maxim jsou u rozostřeného obrazu nižší. Pokud jsou na snímku objekty větší než velikost rozptylové funkce, ostřením se bude měnit pouze velikost spádu na hranách objektu.

Prahovaný výpočet gradientu Jedná se o obdobu postupu výpočtu velikosti gradientu, kdy $n = 1$, $m = 1$,$\Theta \neq 0$. Do proměnné určující výslednou míru zaostření obrázku budeme započítávat pouze příspěvky s větší hodnotou gradientu jasové funkce, které jsou větší než určitá mez.

Výpočet čtverce gradientu Tento postup zvýrazní rozdíly lokálních maxim oproti lokálním minimům gradientu jasové funkce. Tento postup detekuje strmost hrany snímaného objektu a tím pádem kvalita měření neklesá pro větší objekty. Parametry funkce popsané v rovnici (3.1) jsou následující: $n = 1$, $m = 2$, $\Theta = 0$.

Výpočet laplaceova operátoru Parametry pro rovnici (3.1) jsou $n = 2$, $m = 2$, $\Theta = 0$. Ve frekvenční doméně Laplaceův filtr v porovnání s filtry aproximujícími první derivace více zvýrazňuje vyšší frekvence.

Funkce založené na přímém měření jasové funkce

Druhý typ ostřicí funkce $F^2_{f,\Theta}$ dle [8] vychází ze zvolené funkce $\mathbf{f}(z)$, která popisuje výšku vrcholků jasové funkce (*peaks*) a údolí jasové funkce (*valleys*) na snímku (rovnice (3.3)).

$$F^2_{f,\Theta} = \iint_{snímek} \mathbf{f}\Big(I(u,v) - \Theta\Big) \mathrm{d}u \; \mathrm{d}v, \tag{3.3}$$

kde Θ je prahová hodnota a $f(z)$ je funkce blíže specifikovaná v jednotlivých konkrétních příkladech.

Prahování jasových funkcí sekvence snímků Tato metoda sečte prvky sekvence snímků, které se nachází nad stanoveným limitem Θ. Funkce $\mathbf{f}(z)$ odpovídá:

$$\mathbf{f}(z) = \begin{cases} E(z\ ) & \text{pokud } z\ < \Theta, \\ E(-z) & \text{jinak,} \end{cases} \tag{3.4}$$

kde $E(z)$ je definována vzorcem (3.2).

Prahování počtu zaostřených pixelů na sekvenci snímků Tato metoda udává počet pixelů jejichž hodnota jasové funkce je vyšší než určitá mez. V tomto případě funkce $\mathbf{f}(z)$ má následující předpis:

$$\mathbf{f}(z) = \begin{cases} 1 & \text{pokud } z\ > 0, \\ 0 & \text{jinak.} \end{cases} \tag{3.5}$$

Síla signálu Tato metoda počítá sumu čtverců hodnot jasové funkce. V tomto případě funkce $\mathbf{f}(z) = z^2$ a $\Theta = 0$.

Funkce založené na výpočtu odchylky od střední hodnoty jasové funkce

Třetí typ dle [8] ostřicí funkce $F^3_{m,c}$ měří varianci pořízeného snímku.

$$F^3_{m,c} = \frac{1}{c} \iint_{snímek} |I(u,v) - \hat{I}|^m \mathrm{d}u\ \mathrm{d}v, \tag{3.6}$$

Parametr c je normalizující konstanta a $\hat{I}$ značí střední hodnotu intenzity snímku (rovnice (3.6)).

Metoda používající tento popis ostřicí funkce je založena na *rozptylu* hodnot jasové funkce. Parametry funkce (3.6) jsou $m = 2,\ c = A = \iint_{snímek} \mathrm{d}u\ \mathrm{d}v$. Obdobou této metody je použití *normování rozptylu* konstantou $c = A \cdot \hat{I}^2$. Jelikož je výpočet rozptylu náročnou operací je nahrazován výpočtem absolutní hodnoty *variace*. Tedy ostřicí funkce má parametry $F^3_{m=1,c=A}$. Nebo je možné použít tuto metodu v normované variantě, tedy $c = A \cdot \hat{I}$.

Kapitola 4

Začlenění vlastních algoritmů do NI Vision Builderu

V současné době se v rámci projektu používá kamera JAI GO-500M-PGE s telecentrickým objektivem. Pořízená data jsou zpracována a vyhodnocena programem NI Vision Builder (VBAI) ve 32-bitové verzi. Jde o interaktivní aplikaci, pomocí které lze nakonfigurovat kamerové měření (*inspekci*). Výsledný kód je spustitelný jak na počítači s operačním systémem Windows, tak na některém z real-time zařízení, jako jsou inteligentní kamery nebo Vision systémy od National Instruments.

Při analýze programu a nabízených řešení společností National Instruments byly prozkoumány dvě možnosti zapojení vlastního algoritmu do programu VBAI. První je možnost dokoupení licence pro vývojový doplněk programu *Vision Builder AI Development Kit* a následně vytvoření vlastního funkčního bloku programu. Druhou možností je zapojit v rámci funkčního bloku *Run LabVIEW VI* soubor vytvořený v programu LabVIEW.

4.1 Vision Builder AI Development Kit

Tento doplňkový produkt poskytuje možnost návrhu vlastního funkčního bloku s vlastním vzhledem a možností navržení vlastní ikony a uživatelského rozhraní. Informace o tomto modulu lze nalézt v [9].

Vision Builder AI Development Kit předpokládá přístup k následujícím aplikacím

- LabVIEW
- NI Vision Development Module
- NI Vision Acquisition Software
- LabVIEW Real-Time
- Vision Builder AI

Při tvorbě vlastního funkčního bloku programu je možné vybírat z několika šablon.

Simulated Acquisition Step načítá nebo pořizuje snímky a následně je zveřejní pro ostatní procesy pro následné zpracování.

Simple Processing Step zpracovává snímek z předchozího kroku a vrací pouze informaci, zda poskytnutý snímek prošel testem či nikoli.

Processing Step that Logs Measurements provede měření na snímku z předchozího kroku a poskytne naměřené hodnoty následujícím krokům.

Generate a Report poskytuje přístup k měřením z předchozího kroků. Umožňuje úpravu nových kriterií měření dle naměřených dat a vytvoření vlastní podmínky na projití kontroly.

Global Step Status zhodnocuje, zda pořízený snímek prošel veškerými kontrolami v předchozích krocích.

Coordinate System transformuje souřadnice vybrané oblasti vzhledem ke zvolenému souřadnicovému systému.

Tato metoda začlenění vlastního kódu se zdá být výhodná v případě potřeby navržení uživatelského prostředí s nativním vzhledem. Jedná se o placený program a z toho důvodů není zcela vhodný k pokusům bez specifického cíle. Jelikož v rámci řešení bakalářské práce není třeba vytvářet specializovaná uživatelská rozhraní tento přístup nebyl dále zkoumán.

4.2 Připojení souboru LabVIEW VI

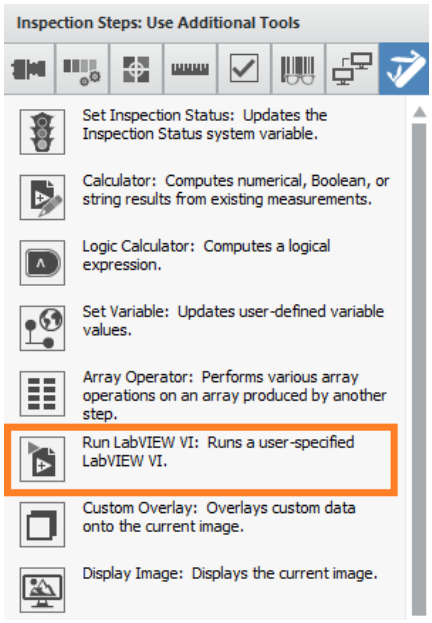
Možnost, která byla prozkoumána podrobně, je připojení souboru LabVIEW (*LabVIEW VI*) pomocí funkčního bloku *Run LABVIEW VI*. Tento blok se nachází v záložce Inspection Steps: Use Additional Tools (obr. 4.1a).

LabVIEW umožňuje připojit externí soubory napsané v různých programovacích jazycích, v závislosti na verzi programu. Ve verzi LabVIEW 2018 je možné volat funkce dynamicky linkované knihovny (*DLL*) psané v jazyce C nebo C++ nebo je možné spustit skrip s příponou *.py* v jazyce Python verze 2.7 nebo 3.5 (v nižších verzích LabVIEW podpora Pythonu není zaručena). V případě, že je třeba využit systémové funkce počítače, lze připojit .NET soubory.

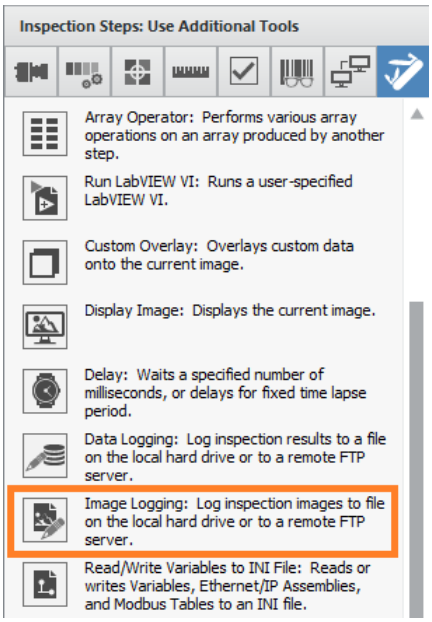
4.2.1 Externí program - Python

Pro ověření funkčnosti a realizaci experimentů byla použita 32 bitová verze Pythonu 2.7. Pro spuštění skriptu v jazyce Python je třeba zapojit tři funkční bloky. Pomocí *Open Session* (obr. A.5a) se vytvoří relace, která poskytne platformu pro spuštění skriptu specifikovanou verzí jazyka Python. Funkční blok *Python Node* (obr. A.5c) volá specifikovaný soubor .py, ve kterém se musí nacházet funkce se názvem uvedeným v (*function name*). Za tento funkční blok je třeba zařadit *Close Python Session* (obr. A.5b), který ukončí relaci s Pythonem.

Pro návrat hodnoty z externího programu slouží proměnná *return value*, u které je nutné definovat její typ. V případě potřeby vracení více návratových hodnot je třeba definovat pole, do nějž budou uloženy návratové hodnoty. Toto



(a) : Blok Run LabVIEW VI se nachází v záložce Use Additional Tools.



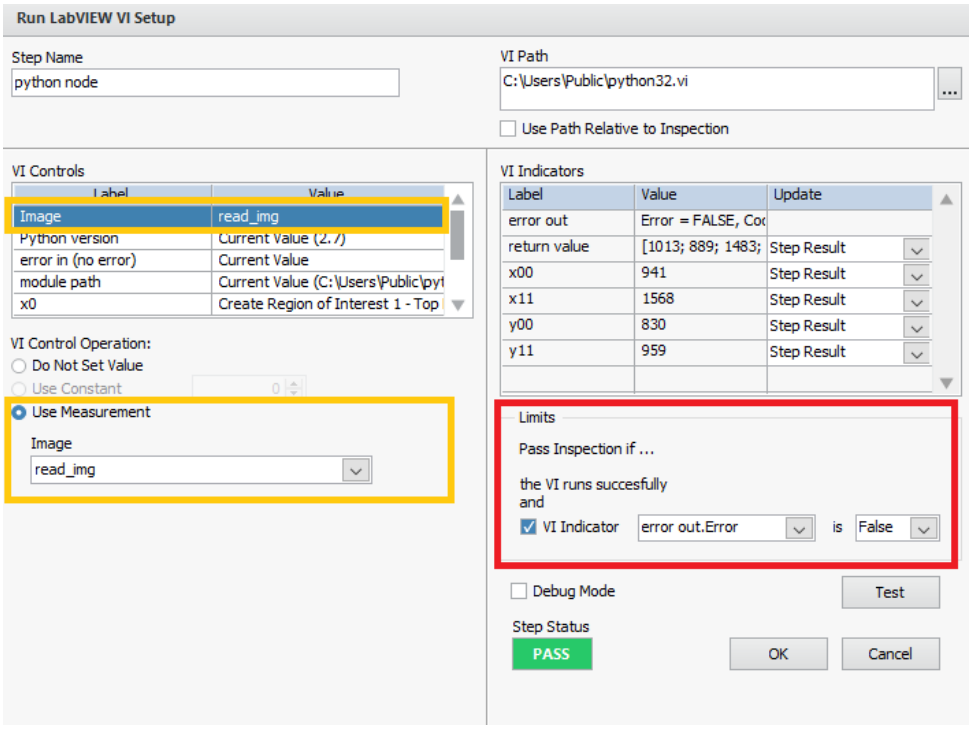
(b) : Blok pro uložení obrázku na disk v počítači.

Obrázek 4.1: Použité funkční bloky pro komunikaci s externími programy v VBAI.

pole může obsahovat pouze prvky stejného typu. Pokud by bylo třeba vracet hodnoty různých datových typů, bylo by potřeba postupovat přes vytvoření souboru, do kterého se uloží hodnoty proměnných, které by následně byly načteny po ukončení běhu skriptu. Příklad souboru LabVIEW VI, který je možné zapojit do VBAI, je zobrazen na obrázku 4.4.

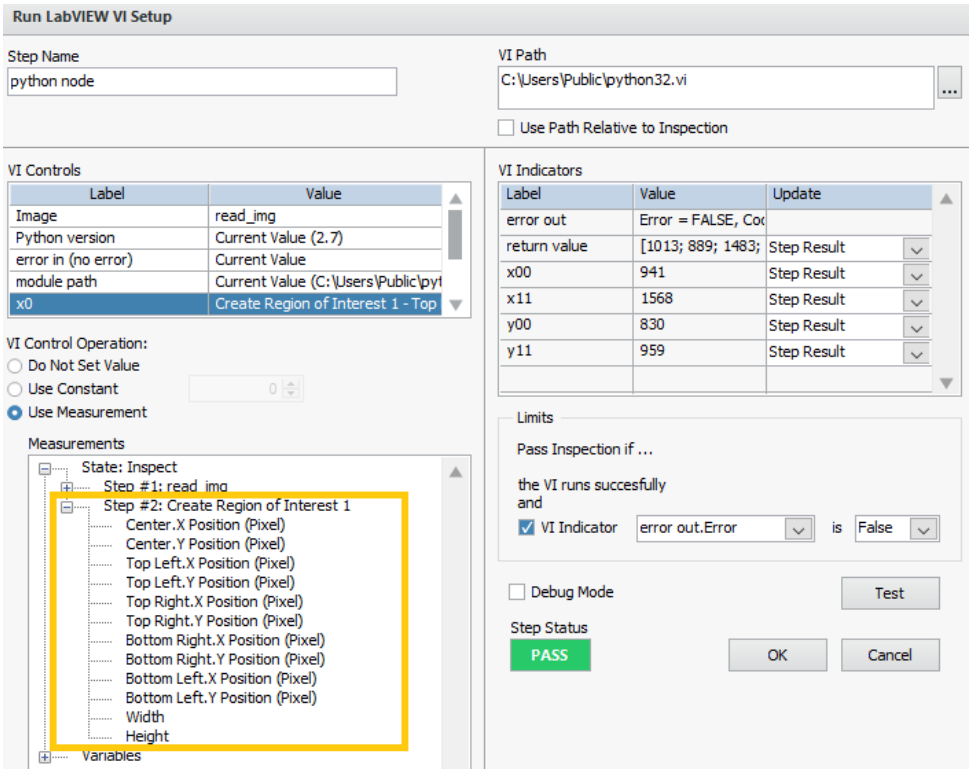
4.2.2 Externí program - DLL

V případě volání knihovny *DLL* je třeba použít blok *Call Library Function Node* (obr. A.6). LabVIEW podporuje volání funkcí konvencí *stdcall* (WINAPI) a konvencí C. V případě, že hlavičkový soubor knihovny DLL neobsahuje pevné celočíselné typy (indikátor typu končící „_t“), je možné využít průvodce pro připojení externí knihovny, který se nachází v záložce *Tools* → *Import* → *Shared Library* (*.dll*). Pro předání černobílého obrázku do knihovny je použit ukazatel na pole hodnot typu „uint8_t“. Proto se knihovna musí připojit ručně. Po přidání bloku *Call Library Function Node* je nutné v záložce *Function* nastavit cestu k .dll souboru, název volané funkce a požadovanou konvenci volání. V záložce *Parameters* je nutné manuálně přidat veškeré vstupní proměnné volané funkce. Příklad nastavení pro předání obrázku ve stupních šedi je na obr. 4.6b. Jelikož se v tomto případě předává pouze ukazatel na začátek pole, je potřeba předat i jeho rozměry. Zjistit počet řádků a sloupců v poli obrázku lze pomocí funkčního bloku *IMAQ GetImageSize* (obr. A.7).



Obrázek 4.2: Nastavení funkčního bloku Run LabVIEW VI. Poté, co je obrázek načten programem VBAI, je možné jej předat jako vstupní parametr volané funkce pomocí volby *Use Measurement*, zobrazeno žlutě. Aby tento funkční blok reagoval na chybu hlášenou externím modulem je třeba nastavit volby zobrazené v červeném obdélníku.

Návratová hodnota tohoto bloku se chová schodně s blokem volajícím skript v jazyce Python. Ovšem pokud je předem známý počet návratových hodnot a jejich typ, lze si je deklarovat v LabVIEW VI a poté přeposlat jejich ukazatele do volané funkce. Není možné deklarovat pole s neznámých počtem prvků, protože deklarace a vyhrazení potřebné paměti probíhá ještě před voláním samotné funkce z *DLL*. Na obrázku 4.5 je uveden příklad blokového diagramu funkce, který je možné volat z VBAI.



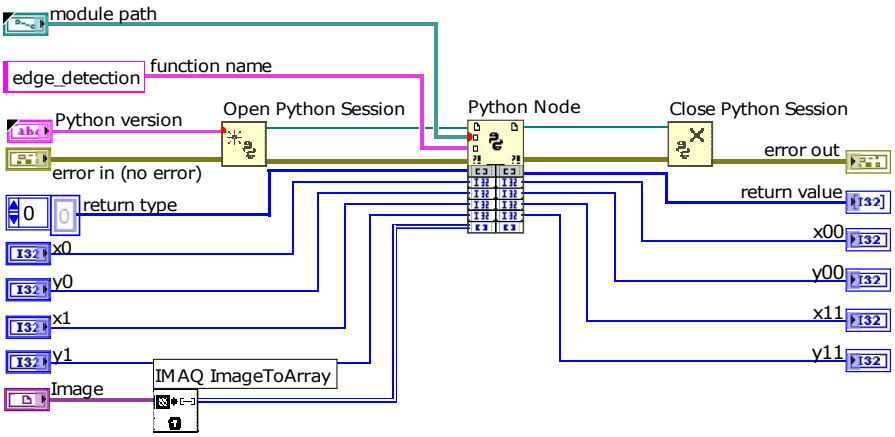
Obrázek 4.3: Předání vybrané oblasti souboru LabVIEW VI. Pomocí funkčního bloku *Create Region of Interest* se vytvoří níže zobrazené proměnné.

4.2.3 Předání snímku

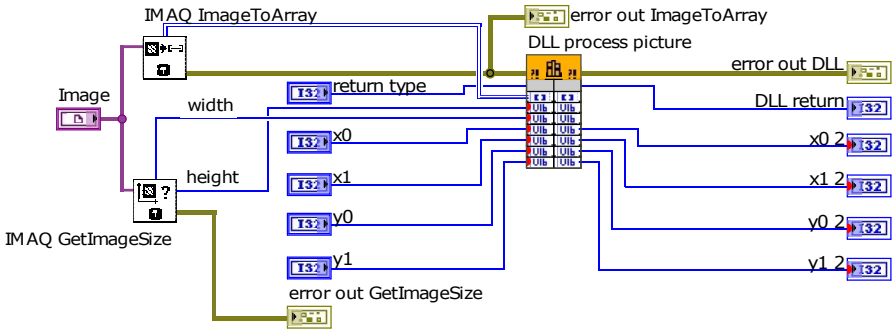
V rámci úlohy hledání hran je důležité zajistit předání snímku mezi LabVIEW VI souborem a VBAI pro jeho následné zpracování. Toho se dá docílit dvěma způsoby. Jednou z možností je předávat data jako proměnné při volání připojeného souboru. Toto lze nastavit v dialogovém okně při změně parametrů bloku Run LabVIEW VI (obr. 4.2). Snímek předaný tímto způsobem se v LabVIEW načte jako typ „*IMAQ Image.ctl*“. Anebo je možné pomocí bloku *Image Logging* (obr. 4.1b) uložit snímek do souboru ve formátu JPEG, BMP, TIFF. Na straně LabVIEW by soubor byl načten blokem *IMAQ ReadFile2* (obr. A.3).

Výběr oblasti zájmu na snímku je možné provést pomocí funkčního bloku *Create Region of Interest*. Následně lze předat parametry vybrané oblasti jako parametry volané funkce. Příklad předání parametrů je zobrazen na obrázku 4.3.

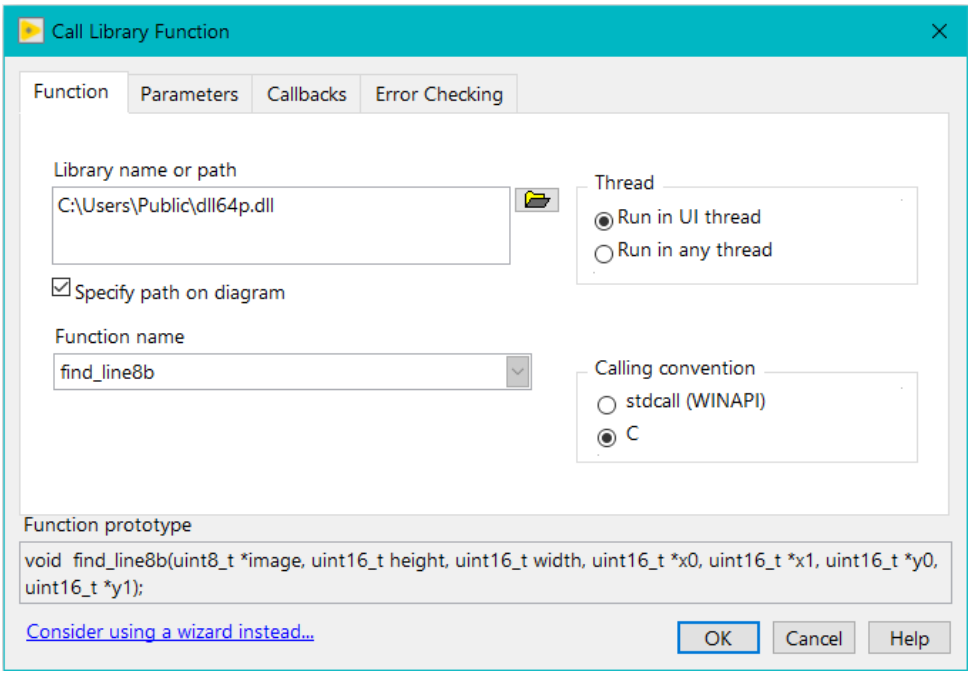
Po inicializaci obrázku, buď jeho předáním jako proměnné nebo načtením pomocí funkčního bloku, je třeba jej převést na pole číselných hodnot, což se zajistí pomocí funkce *IMAQ ImageToArray* pro černobílý obrázek, nebo *IMAQ ColorImageToArray* pro barevný obrázek (viz obr. A.4).



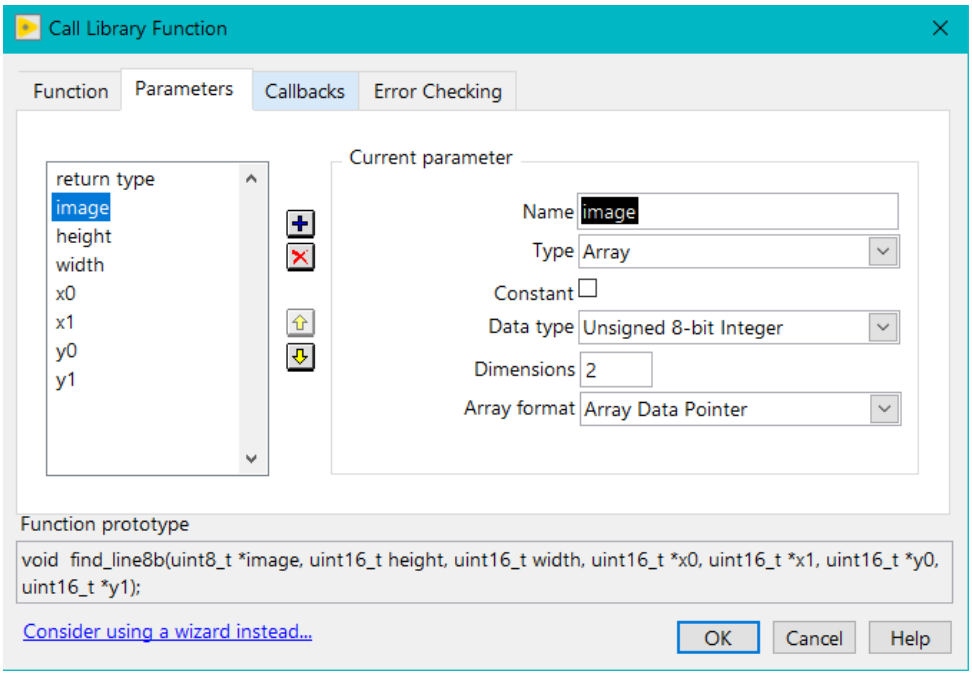
Obrázek 4.4: Příklad blokového diagramu volání Python skriptu na úpravu snímku. *Module path* obsahuje cestu k .py souboru, v němž se nachází funkce se jménem *function name*. Použitá verze Pythonu je uvedena v *Python version*. V rámci konvence používané v LabVIEW je třeba provést deklaraci návratové hodnoty externí funkce. Proměnné x0, y0, x1, y1 reprezentují rohy vybrané oblasti snímku. Do proměnné Image je z VBAI načten snímek.



Obrázek 4.5: Příklad blokového diagramu volání knihovny *DLL*. Předávají se ukazatele na deklarované proměnné. V případě pole, je třeba udat i jeho rozměry, ty je možné zjistit pomocí funkčního bloku *IMAQ GetImageSize*.



(a) : Záložka Function.



(b) : Záložka Parameters. Parametry jsou nastaveny pro předání obrázku ve stupních šedi.

Obrázek 4.6: Nastavení volání externí knihovny *DLL*.

Kapitola 5
Návrh algoritmu hranové detekce

Pro zjištění pozice součástky se využívají buď přímo hrany tvořené vrstvami na substrátu, nebo hrany zrcátka či jiné součástky na něm umístěné. V obou případech se vyskytují známé vady viditelných hran. Při obrábění zrcátka nebo součástky diamantovými kotoučky vznikají úlomky, které se projevují jako nerovnosti hrany ve směru do plochy součástky. Při nanášení jednotlivých vrstev na substrát, dochází k nedokonalému dosednutí masek a můžou vznikat vícenásobné hrany, které jsou tvořeny různými technologickými vrstvami. Z důvodu požadavku určení přesné polohy se nabízí možnost použití algoritmu pro subpixelové určení polohy.

Pro detekci a lokalizaci hran byl navržen vlastní algoritmus, který lze rozdělit do několika kroků. V prvním kroku je obrázek filtrován, poté je zjištěná poloha potenciálních hranových bodů a směr derivace jasové funkce v nich. Následně jsou potenciální hranové body vybrány podle zadaných kritérií a jejich poloha je upřesněná pomocí aproximace S-křivkou. Výsledné body jsou proloženy přímkou, která je analytickým modelem hledané hrany.

5.1 Filtrace a hledání hranových bodů

V prvním kroku je na snímek aplikován Gaussův filtr velikosti 3×3 pixely, $\sigma = 1$, který zajišťuje redukci šumu. Poté je zjištěna poloha hranových bodů Cannyho hranovým detektorem s velikostí masky 3×3 pixely. Horní mez (t_h) a dolní mez (t_d) pro prahování je určena dle vzorců (5.1) a (5.2).

$$t_h = \min\{255;\ 1,33\ \tilde{I}\}, \tag{5.1}$$

$$t_d = \frac{t_h}{3}, \tag{5.2}$$

kde $\tilde{I}$ je medián jasové funkce snímku.

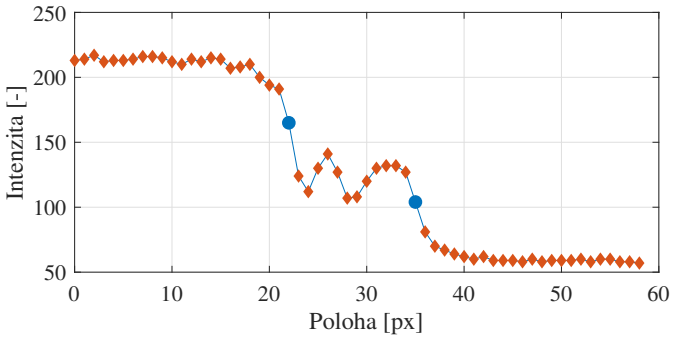
Směr derivace v těchto bodech je určen pomocí aproximace gradientu v osmi směrech Sobelovým operátorem (tabulka 2.1). Tím jsou získány potenciální hranové body a odpovídající gradienty.

5.2 Výběr hranových bodů

Z potenciálních hranových bodů jsou následně vyřazeny ty, jejichž gradient neodpovídá očekávanému směru jasového přechodu na snímku, směr přechodu volí uživatel.

Také jsou vyřazeny body, jejichž gradient se liší od normály očekávané hrany o více jak 45 stupňů. Toto omezení bylo použito, protože uživatel přibližně zarovnává substrát s osami snímku. Limity na směr normály vychází z nejistoty zarovnání a detekce gradientu v osmi směrech.

U snímku, kde na vzoru substrátu vznikají vícenásobné hrany (obr. 5.1), může vzdálenost těchto hran přesáhnout 5 pixelů a hranový detektor je proto detekuje jako samostatné hrany. Proto byl implementován algoritmus, který zjistí, zda daný hranový bod je prvním detekovaným hranovým bodem ve směru přechodu z černé do bílé. Ostatní potenciální hranové body jsou vyřazeny.



Obrázek 5.1: Graf zobrazuje řez jasové funkce ve směru největšího spádu při přechodu z bílé do černé. Červené body označují naměřené hodnoty jasové funkce. Modré body značí detekované potenciální hranové body.

5.3 Aproximace S-křivkou

Pro každý hranový bod je vybráno k bodů z 8-okolí ve směru gradientu na obě strany. Gradient je definován v osmi diskrétních směrech. Při experimentech bylo použito 20 bodů ($k = 20$). Tato hodnota byla určena s ohledem na rozlišení snímku a profil obvyklé hrany tak, aby byly v takto vzniklém okně body i z okolí hrany (viz. obr. 5.2).

Vybrané body jsou následně proloženy S-křivkou (kapitola 2.2.3). Z parametrů křivky je získán nový přesněji lokalizovaný hranový bod.

Hodnocení kvality aproximované S-křivky

V případě, že se podařilo proložit naměřená data požadovanou S-křivkou $S(x)$ (dle rovnice (2.12)), je třeba rozhodnout, zda se jedná o platnou aproximaci.

$$S(x) = \frac{L}{1 + e^{-k(x-x_0)}} + b.$$

Kromě klasické metody zjištění platnosti aproximace (R-squared), lze v našem specializovaném případě použít omezení parametrů funkce. Počítejme, že snímek není saturovaný, tedy parametr L musí nabývat hodnot menších jak 255. Na parametr x_0 bylo aplikováno omezení, které říká, že poloha odhadovaného

inflexního bodu nemá být posunuta o více jak 2 pixely. Je zřejmé, že parametr b nemá být větší jak 254, jelikož není smysluplné aproximovat křivku mimo měřitelné hodnoty. Tyto parametry jsou obecně platné pro optická měření černobílou kamerou s rozsahem jasu 0-255 (8 bitů).

Při pokusech prováděných v této bakalářské práci se dají tyto omezení zpřísnit. Z jasových funkcí pořízených snímku lze usoudit, že parametr b by neměl nabývat hodnot větších jak 80. Tento parametr je ovlivněn nejnižší hodnotou jasové funkce v aproximované oblasti. Parametr L je shora omezen vztahem $L < 255 - b$. Pokud aproximovaná křivka nesplňuje tyto požadavky je příslušný inflexní bod vyřazen z dalšího zpracování.

5.4 Lokalizace hrany

Hranové body získané v předchozím kroku jsou následně proloženy analytickou křivkou reprezentující hranu. V této bakalářské práci jsou hranové body aproximovány přímkou, protože dobře odpovídá tvaru součástek i textuře na substrátu. Při návrhu a testování algoritmu byla vyzkoušena metoda nejmenších čtverců, vážených nejmenších čtverců a metoda RANSAC.

5.4.1 Nejmenší čtverce a vážené nejmenší čtverce

Algoritmus s aproximací nejmenšími čtverci se ukázal jako méně vhodný. Z toho důvodu byla otestována metoda vážených nejmenších čtverců. Hranovým bodům byla přiřazena váha w_i

$$w_i = \frac{L_i}{2} \tan\left(0,2455\ k_i\right), \tag{5.3}$$

kde $L_i,\ k_i$ jsou parametry S-křivky definované vztahem (2.12).

I přímka získaná metodou vážených nejmenších čtverců je zatížena chybou, kterou způsobují často detekované odlehlé body. Proto byla vyzkoušena varianta prokládání hranových bodů váženou křivkou. V tomto případě byl proveden první odhad přímky pomocí metody vážených nejmenších čtverců. Poté je vypočtena míra kvality odhadu (cena přímky) jako součet příspěvků hranových bodů do jednotlivých bodů přímky. Hranový bod x_{0i} (inflexní bod dle S-křivky) přispívá do ceny pixelu dle rovnice (5.4).

$$c(x_n, x_{0i}) = \begin{cases} 1 - \dfrac{|x_n - x_{0i}|}{x_{di}} & \text{pokud } |x_n - x_{0i}| < x_{di}, \\ 0 & \text{jinak,} \end{cases} \tag{5.4}$$

kde proměnná x_n vyjadřuje polohu zkoumaného pixelu. Velikost parametru x_{di} je vypočtena podle rovnice

$$x_{di} = \frac{L_i}{2} \tan(0,2455\ k_i), \tag{5.5}$$

kde $L_i,\ x_{0i},\ k_i$ jsou parametry S-křivek definované vztahem (2.12). Tedy celková cena v_n jednotlivých pixelů (rovnice (5.6)) je součtem příspěvků

jednotlivých inflexních bodů každému z nich.

$$v_n = \sum_i c(x_n, x_{0i}), \tag{5.6}$$

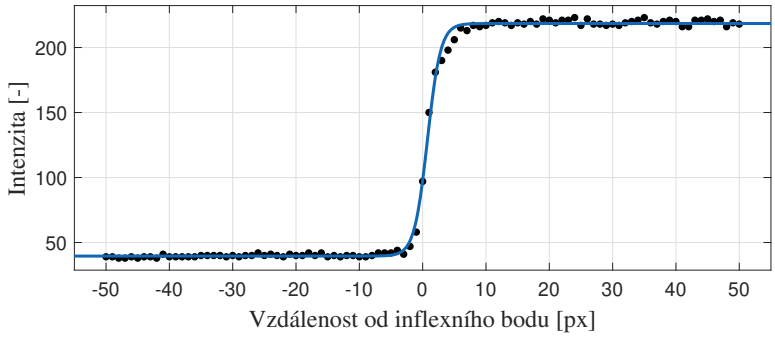
kde x_{0i} je poloha i-tého inflexního bodu a x_n je poloha pixelu, kterým prochází proložená přímka. Cena přímky C je určena součtem cen pixelů jimiž prochází

$$C = \sum_n v_n. \tag{5.7}$$

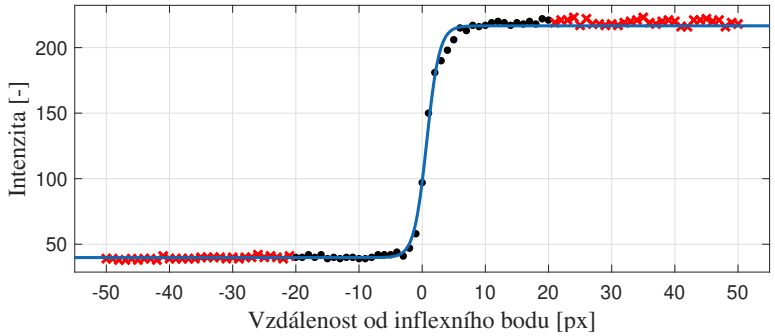
Po spočtení ceny přímky byly vyřazeny body ležící o více jak 4 pixely od aproximované přímky ve světlé části zrcátka. Následně byly nevyřazené hranové body znovu aproximovány. Tento postup se opakoval dokud rostla cena proložené přímky. I tato metoda se ukázala jako nedostatečně spolehlivá.

5.4.2 RANSAC

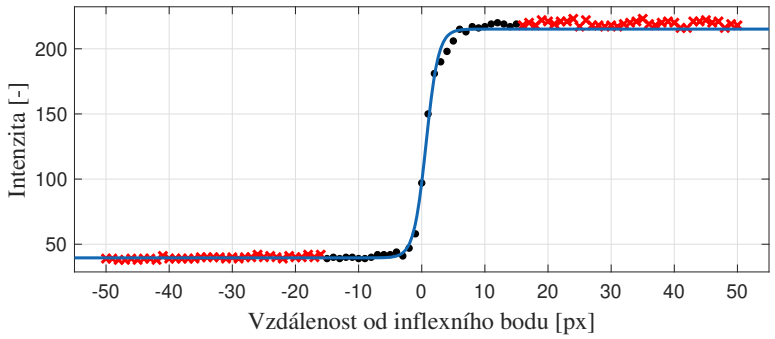
Metoda implementovaná ve výsledném algoritmu je metoda RANSAC. Byla vybrána pro její schopnost vyřadit odlehlé hodnoty. Nejdříve jsou nalezeny *inliers* hodnoty algoritmem RANSAC. V jedné iteraci tohoto algoritmu byly aproximovány přímkou vždy dva náhodně vybrané hranové body získané z aproximace S-křivkou. Z testů vyplývá, že průměrně je detekováno 20% odlehlých hranových bodů. Ukázalo se, že pro opakovatelnost výsledku, kdy aproximovaná hrana se skládá z 1000 hranových bodů, je třeba 300 iterací. Výsledné parametry přímky jsou získány proložením *inliers* hodnot pomocí metody nejmenších čtverců.



(a) : Pro prokládání bylo vybráno 50 bodů ve směru a 50 bodů proti směru gradientu od potenciálního inflexního bodů. Aproximovaná funkce je $S(x) = \frac{178,90}{1+e^{-0,97(x-0,76)}} + 39,56$.



(b) : Pro prokládání bylo vybráno 20 bodů ve směru a 20 bodů proti směru gradientu od potenciálního inflexního bodů. Aproximovaná funkce je $S(x) = \frac{176,98}{1+e^{-1,00(x-0,72)}} + 39,82$.



(c) : Pro prokládání bylo vybráno 15 bodů ve směru a 15 bodů proti směru gradientu od potenciálního inflexního bodů. Aproximovaná funkce je $S(x) = \frac{175,60}{1+e^{-1,027(x-0,69)}} + 39,54$. Jedná se o mezný počet bodů, při menším počtu prokládání křivkou často selhává.

Obrázek 5.2: Černé body reprezentují prokládané hodnoty. Červené křížky označují body vyřazené z prokládání.

Kapitola 6

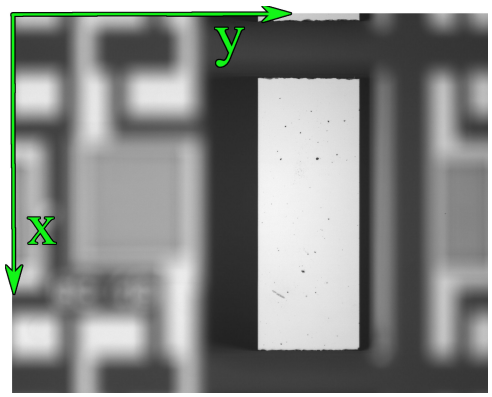
Ověření funkčnosti a testování

Funkčnost a porovnání algoritmů jsou dokumentovány dvěma experimenty. První experiment je prováděn na sekvenci snímků zaznamenávajících posun scény v rovině snímku. Druhý experiment dokumentuje vliv natočení snímané scény na detekci hrany. Jako scéna pro prezentaci byla vybrána oblast substrátu obsahující zrcátko. Obrázek je zaostřen na jeho horní stěnu, kterou na snímku tvoří světlý obdélník. Scéna je na obrázku 6.1. Testovány byly čtyři algoritmy. Původní algoritmus implementovaný v NI Vision Builder (značeno *VBAI*) a navržený algoritmus s metodami prokládání hranových bodů nejmenšími čtverci (značeno *LSQ*), váženými nejmenšími čtverci (značeno *WLSQ*), popsané v kapitole 5.4.1, a metodou RANSAC (značeno *RANSAC*), popsánou v kapitole 5.4.2.

Pomocí snímku kalibru byla stanovena hodnota v mikrometrech odpovídající jednomu pixelu

$$1\text{px} = (0,57 \pm 0,02)\mu m.$$

V této kapitole je použit souřadnicový systém s počátkem v levém horním rohu, ze kterého osa x směřuje dolů a osa y směřuje doprava (obr. 6.1).

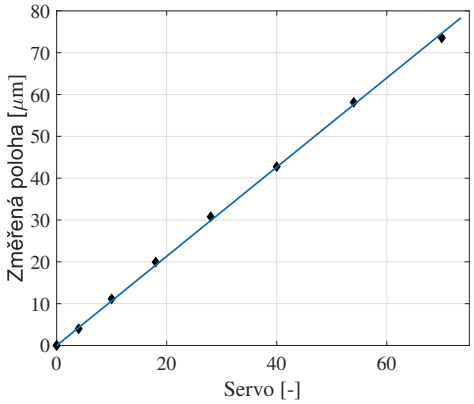


Obrázek 6.1: Použitá scéna s vyznačeným souřadnicovým systémem.

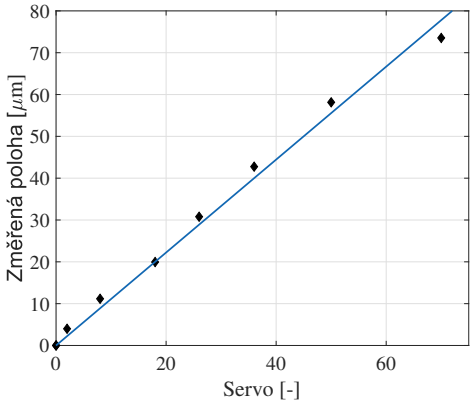
6.1 Experiment s translací scény

První experiment vychází z dat, které vznikly pořízením sekvence snímků scény, která se posouvá. Posun zajišťuje motoricky poháněný polohovací stolek. Řídící systém zajišťuje řízení na polohu. Byly pořízeny sekvence pěti snímků v osmi polohách, které se posouvali ve směru osy y (dle použitého souřadnicového systému ve snímcích). Poloha scény byla zvolena [0, 4, 10, 18, 28, 40, 54, 70] pixelů od počáteční polohy. Výsledky detekce jsou uvedeny v tabulce 6.1 a jednotlivé posuny vůči čítačům jsou zobrazeny v grafech na obrázku 6.2.

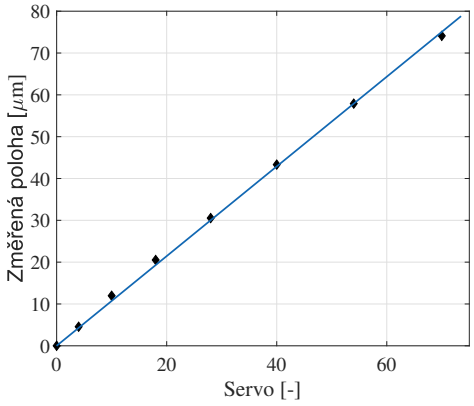
Polohovací stolek používá rozhraní, přes které je možné zadat požadovanou polohu. Po obdržení požadavku na změnu polohy se vykoná tento pohyb a ze senzoru umístěného na servomotoru je odeslána informace o dosažené poloze. Levé grafy na obrázku 6.2 ukazují požadovanou polohu po polohovacím stolku a pravé grafy ukazují polohu dle servoregulátoru.



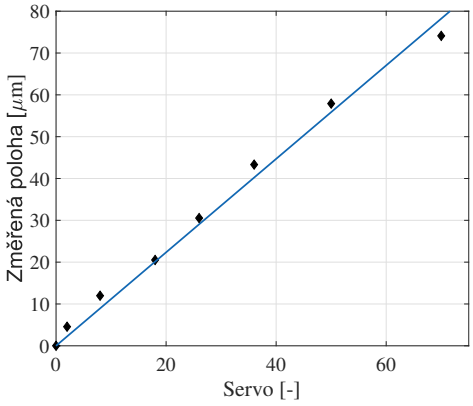
(a) : RANSAC, požadovaná poloha. Proloženo přímkou $y = 1,07x$.



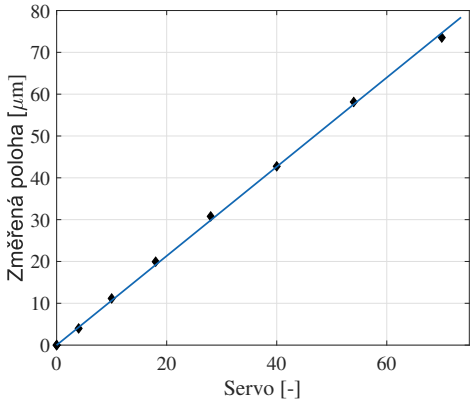
(b) : RANSAC, změřená poloha senzorem. Proloženo přímkou $y = 1,11x$.



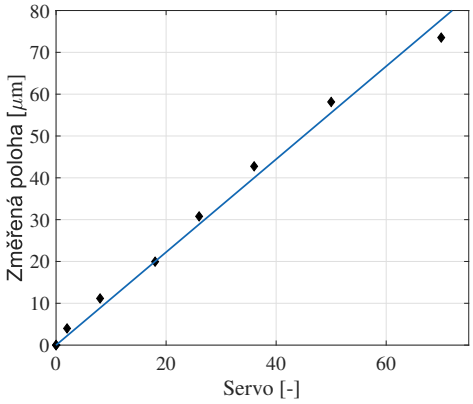
(c) : LSQ, požadovaná poloha. Proloženo přímkou $y = 1,08x$.



(d) : LSQ, změřená poloha senzorem. Proloženo přímkou $y = 1,12x$.



(e) : WLSQ, požadovaná poloha. Proloženo přímkou $y = 1,07x$.



(f) : WLSQ, změřená poloha senzorem. Proloženo přímkou $y = 1,11x$.

Obrázek 6.2: Porovnání tří metod prokládání hranových bodů implementovaných v navrženém algoritmu.

	Očekávaný posun [px]	Skutečný posun [px]	VBAI [μm]	RANSAC [μm]	LSQ [μm]	WLSQ [μm]
Počátek [px]	-	-	1469.73	1470.36	1468.94	1470.36
	4	0	4.20	3.99	4.80	4.31
	10	1	11.76	11.16	12.21	13.18
	18	9	20.19	19.95	20.76	27.93
Posun	28	17	30.75	30.78	30.78	30.21
	40	27	42.98	42.75	43.56	42.18
	54	37	58.13	58.14	58.14	58.45
	70	69	73.66	73.53	74.34	72.96
Konec [px]	-	-	1598.95	1599.36	1599.36	1599.36

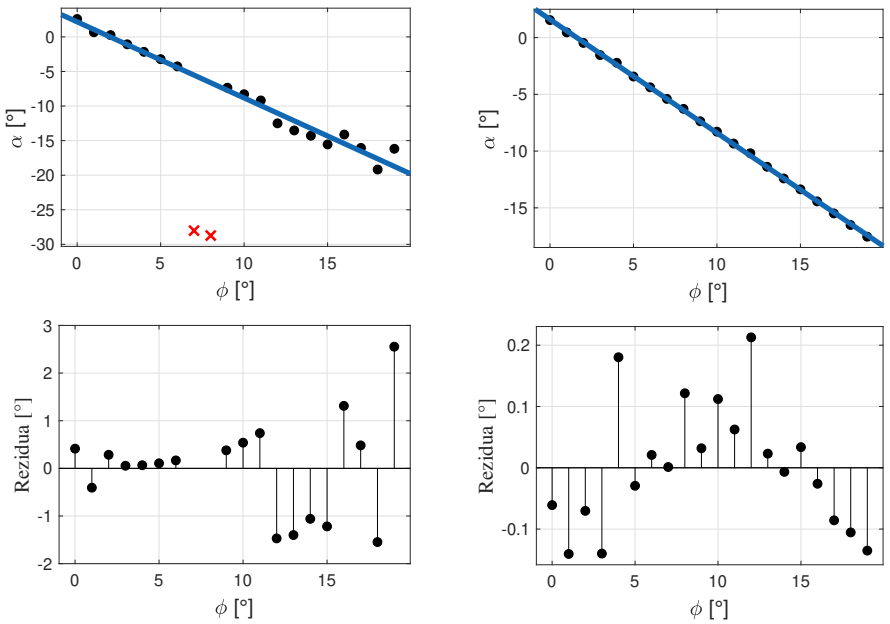
Tabulka 6.1: Tabulka hodnot posunu bodu určeného protnutím přímky aproximované různými metodami s osou y.

6.2 Experiment s rotací scény

V rámci porovnání úspěšnosti implementovaného algoritmu využívajícího prokládání RANSAC a algoritmu VBAI byly provedeny pokusy s rotací scény. V době pořizování dat nebylo možné provést fyzickou rotaci vzorku v rovině substrátu. Experiment byl proveden pořízením 20 snímků v konstantní poloze a následnou rotací o jeden stupeň v programu Gimp s lineární interpolací. Snímaná scéna je zaostřena na horní plochu zrcátka, které má na sobě znatelné vady a je zašpiněna prachem. Výsledky hledání hrany jsou zobrazeny v tabulce 6.2, která udává úhel α svíraný osou y a detekovanou přímkou. Pro srovnání je uvedena rotace snímku (ϕ), na kterém je prováděna hranová detekce, oproti pořízenému snímku. Rozdíly jsou způsobeny počátečním zarovnáním scény. V grafech na obrázku 6.3 jsou zobrazeny proložení průběhů měření jednotlivými algoritmy. Algoritmus implementovaný v programu NI Vision Builder při rotaci ϕ =7° a ϕ =8°detekoval přímku zcela chybně, proto byly naměřené hodnoty příslušné těmto úhlům vyloučené z prokládání.

ϕ [°]	VBAI [°]	RANSAC [°]
0	2,59	1,54
1	0,67	0,46
2	0,26	-0,47
3	-1,07	-1,54
4	-2,16	-2,22
5	-3,22	-3,43
6	-4,26	-4,38
7	-28,02	-5,40
8	-28,73	-6,28
9	-7,35	-7,37
10	-8,29	-8,29
11	-9,19	-9,34
12	-12,50	-10,19
13	-13,53	-11,38
14	-14,29	-12,41
15	-15,55	-13,37
16	-14,12	-14,43
17	-16,05	-15,49
18	-19,18	-16,51
19	-16,18	-17,54

Tabulka 6.2: Tabulka úhlů svíraných přímkami prokládané hrany s osou y v závislosti na rotaci scény o úhel ϕ.



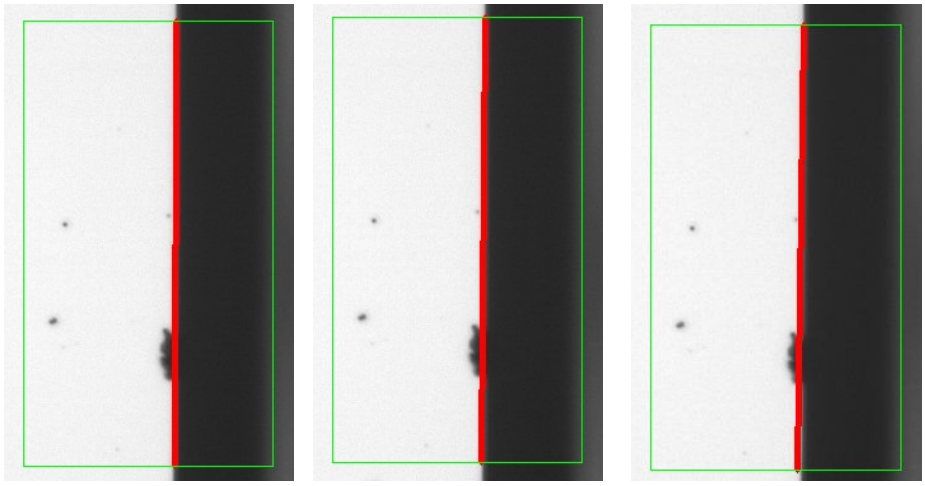
(a) : VBAI,
Hodnoty jsou proloženy přímkou $y = -1,10\ x + 2,18$.

(b) : RANSAC,
Hodnoty jsou proloženy přímkou $y = -x + 1,60$.

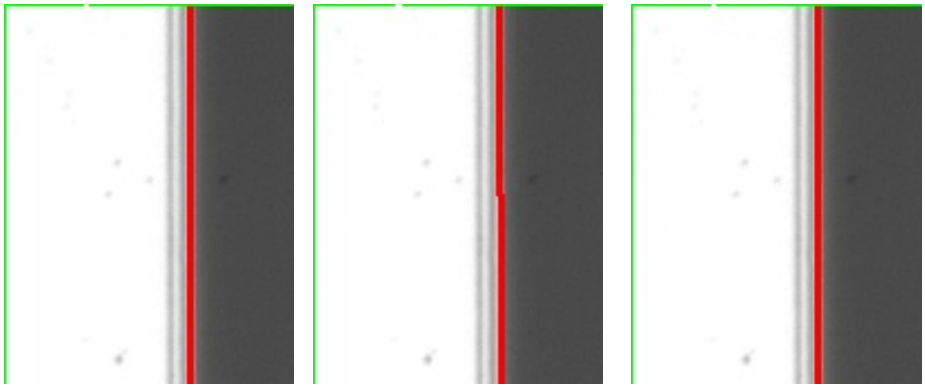
Obrázek 6.3: Grafy zobrazují proložení naměřených úhlů α, které svírají detekované přímky s osou y na snímcích, v experimentu s rotací scény o ϕ stupňů. Červeně jsou značeny body vyloučené z prokládání naměřených hodnot.

6.3 Porovnání verzí navrženého algoritmu

Navržený algoritmus byl testován se třemi metodami aproximace hranových bodů přímkou. V první variantě byla vyzkoušena aproximace bodů metodou nejmenších čtverců. Poté byla otestována metoda vážených nejmenších čtverců, kdy prokládaným bodů byla přidána váha dle rovnice (5.3). Následně byla implementována metoda RANSAC. Tyto tři metody proložení byly testovány na vybraných datech s výskytem prachu na snímku a na snímcích se součástkami s poškozenými (nepravidelnými) okraji. Na těchto snímcích se metoda RANSAC ukázala být výhodnou pro detekci hrany, protože vyřadí detekované odlehlé hranové body vzniklé vadami snímané scény. Obrázky 6.4, 6.5 zobrazují vybrané scény na nichž se projevuje rozdíl v algoritmech.

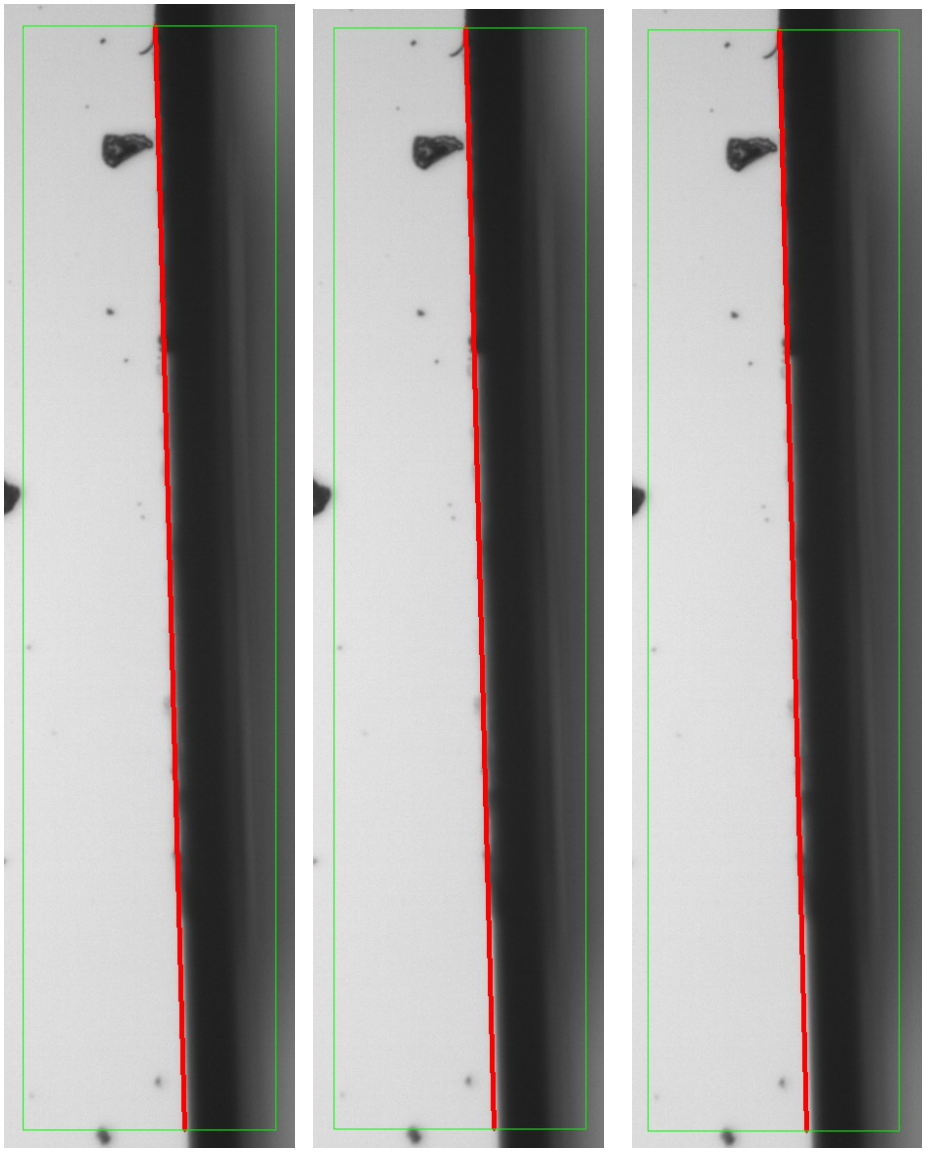


(a) : RANSAC, $y = 1466,82.$ **(b) :** WLSQ, $y = -0,01x + 1471,71.$ **(c) :** LSQ, $y = -0,01x + 1476,86.$



(d) : RANSAC, $y = +82,90.$ **(e) :** WLSQ, $y = 0,01x + 83,00.$ **(f) :** LSQ, $y = 83,00.$

Obrázek 6.4: Porovnání výsledků hledání hrany algoritmem RANSAC, WLSQ a LSQ ve vybrané oblasti. Snímky a) až c) reprezentují typickou hranu zrcátka. Snímky d) až f) reprezentují vícenásobnou hranu.



(a) : RANSAC, $y = 0,03x + 1674,44$.

(b) : WLSQ, $y = 0,03x + 1675,52$.

(c) : LSQ, $y = 0,02x + 167592$.

Obrázek 6.5: Porovnání výsledků hledání hrany algoritmem RANSAC, LSQ, WLSQ ve vybrané oblasti. Snímky reprezentují hranu součástky (zrcátka) jejíž povrch není zcela čistý.

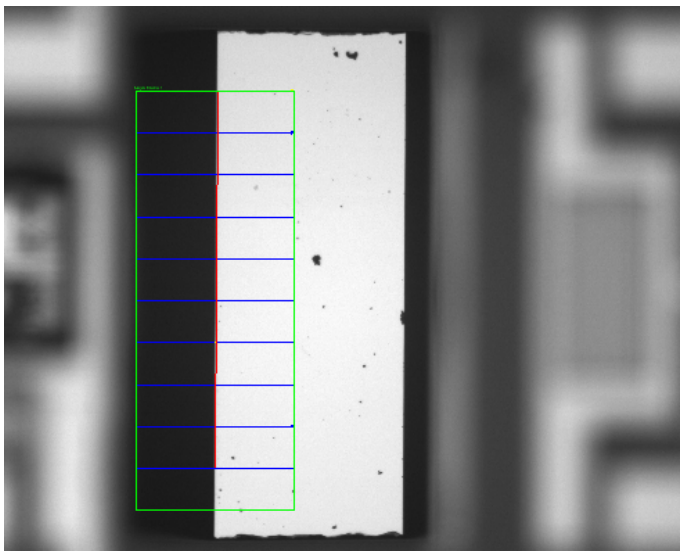
6.4 Vyhodnocení výsledků

Jak je vidět z tabulky 6.1, posun zadaný do řídícího systému neodpovídá provedenému posunu. Tento jev vznikl nejspíše z důvodu nezkorigování časování pořízení snímku a časování odpovědí řídicího systému. Servomotor je schopný dosáhnout polohy s přesností $1\mu m$. Servosystém je nastaven na identifikaci prvního průchodu požadovanou polohou v dané přesnosti. Je pravděpodobné, že poté, co systém ohlásí dosažení cílové polohy, se stále pohybuje. Při příštím měření bude potřeba po ohlášení konce pohybu servosystémem sečkat několik vteřin a poté pořídit snímek scény.

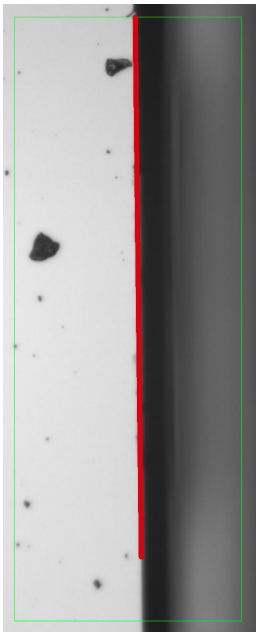
U všech navržených algoritmů je detekovaná hrana posunutá oproti hraně detekované VBAI z důvodu použití většího počtu hranových bodů a následnému zpřesnění jejich polohy pomocí aproximace S-křivkou. Při detekci hrany programem VBAI byla pozice hrany detekována pouze v devíti řezech (obr. 6.6). Všechny algoritmy dávají stabilní a opakovatelné výsledky, v případě, že snímek neobsahuje znatelné vady výrobku.

V experimentech s rotací při hledání hrany zrcátka algoritmus VBAI v některých případech určil prokládanou hranu chybně. Na obrázcích 6.7 a 6.8 je vidět, že při určitém úhlu natočení hledané hrany, jsou programem VBAI upřednostňovány výrazné černé body na ploše součástky (prachové částice).

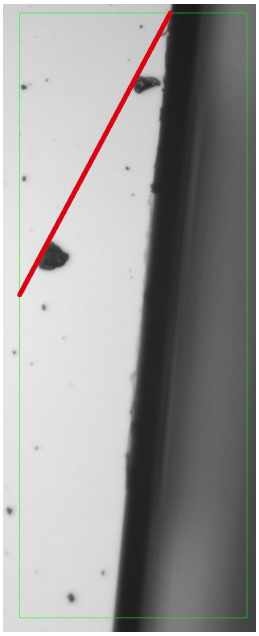
Ze získaných dat vyplývá, že při silnějším zašpinění zrcátka prachem a větším počtu vad na hranách je výhodné použit metodu RANSAC. Tato metoda dokáže omezit vliv těchto vad scény a poskytuje přesnější údaje.



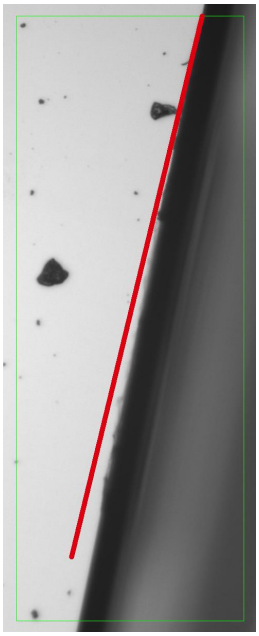
Obrázek 6.6: Modře jsou označeny vybrané řezy snímku, které program VBAI využívá pro určení polohy hrany.



(a) : Rotace scény o 1°. $\alpha = 0{,}67°$.

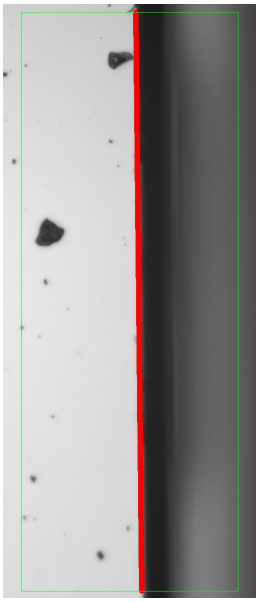


(b) : Rotace scény o 7°. α = -28,02°.

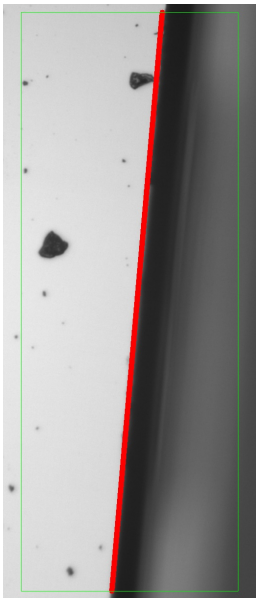


(c) : Rotace scény o 13°. α = -13,53°.

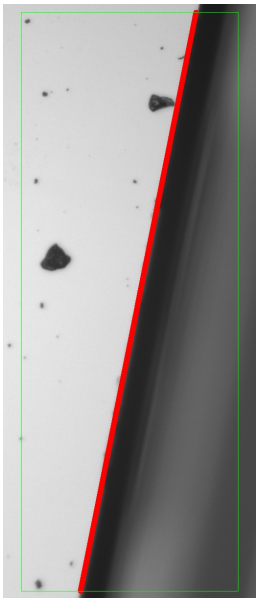
Obrázek 6.7: Příklady detekce směru hrany algoritmem VBAI. Kde α je úhel sevřený detekovanou přímkou s osou y.



(a) : Rotace scény o 1°. $\alpha = 0{,}46°$.



(b) : Rotace scény o 7°. α = -5,40°.



(c) : Rotace scény o 13°. α = -11,38°.

Obrázek 6.8: Příklady detekce směru hrany algoritmem RANSAC. Kde α je úhel sevřený detekovanou přímkou s osou y.

Kapitola 7

Závěr

V rámci této bakalářské práce jsem navrhla algoritmus pro lokalizaci hran v obraze, který je základem pro přesné polohování. Algoritmus vyhledá hranové body odpovídající detekované hraně a poté je proloží analytickou křivkou, která v této aplikaci představuje přímku. Vyzkoušela jsem tři metody prokládání hranových bodů přímkou, metodu nejmenších čtverců, metodu vážených nejmenších čtverců a RANSAC.

Nalezla a vyzkoušela jsem možnosti začlenění vlastního algoritmu do programu NI Vision Builder. Z nalezených postupů byla využita možnost připojení externích modulů jako vlastního kroku inspekce programu NI Vision Builder pomocí funkčního bloku Run LabVIEW VI. Programovatelný blok spouští VI soubor, který obsahuje externí algoritmy zapsané ve skriptovacím jazyce Python, nebo volá funkce externí knihovny DLL. Algoritmus byl mnou implementován v jazyce Python 2.7-32bit.

Navržený algoritmus jsem porovnala s dostupnou funkcí implementovanou v NI Vision Builder na hledání hran. Ukázalo se, že pro snímky s malým množstvím prachu a vad hrany dosahuje navržený algoritmus srovnatelných výsledků jako výchozí funkce v NI Vision Builder. Pro snímky s větším počtem prachových částic nebo pro snímky, na nichž jsou nerovné hrany součástek (způsobeno výrobou), je navržený algoritmus s metodou RANSAC přesnější. Tento algoritmus dokázal účinně omezit vliv chybně detekovaných hranových bodů na polohu hrany.

Literatura

[1] ŠONKA, Milan. *Image Processing, Analysis and Machine Vision*. London: Chapman and Hall Computing, 1993. ISBN 0-412-45570-6.

[2] FISCHLER, Martin A. a Robert C. BOLLES. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*. 1981, **24**(6), 381-395. DOI: 10.1145/358669.358692. ISSN 00010782.

[3] KRAUS, Svatopluk. *Měření rozměrů CCD kamerou se submikronovou přesností*. Praha: ČVUT, FEL, 1999. Disertační práce. ČVUT.

[4] SCHARR, Hanno. *Optimal operators in digital image processing*. Dostupné také z: https://www.researchgate.net/publication/36148383_Optimal_operators_in_digital_image_processing_Elektronische_Ressource. Heidelberg, Germany, 2000. Disertační práce. Ruprecht-Karls-Universität.

[5] HARALICK, Robert M. A Facet Model for Image Data: Regions, Edges, and Texture. *Digital Image Processing*. Dordrecht: Springer Netherlands, 1981, 1981, (15), 337-356. DOI: 10.1007/978-94-009-8543-8_20. ISBN 978-94-009-8545-2.

[6] CANNY, John. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1986, **PAMI-8**(6), 679-698. DOI: 10.1109/TPAMI.1986.4767851. ISSN 0162-8828.

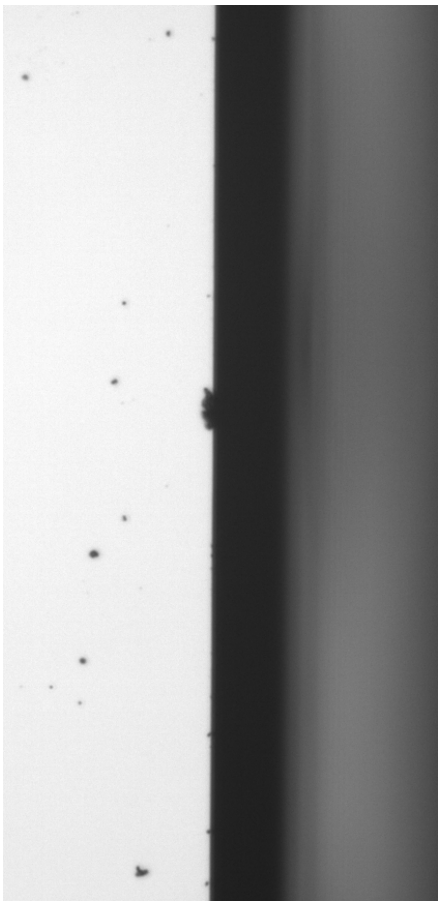
[7] BALTAG, Octavian. History of Automatic Focusing Reflected by Patents. *Science Innovation*. 2015, **3**(1), 1-17. DOI: 10.11648/j.si.20150301.11. ISSN 2328-7861.

[8] GROEN, Frans C. A., Ian T. YOUNG a Guido LIGTHART. A comparison of different focus functions for use in autofocus algorithms. *Cytometry*. 1985, **6**(2), 81-91. DOI: 10.1002/cyto.990060202. ISSN 0196-4763. Dostupné také z: http://doi.wiley.com/10.1002/cyto.990060202

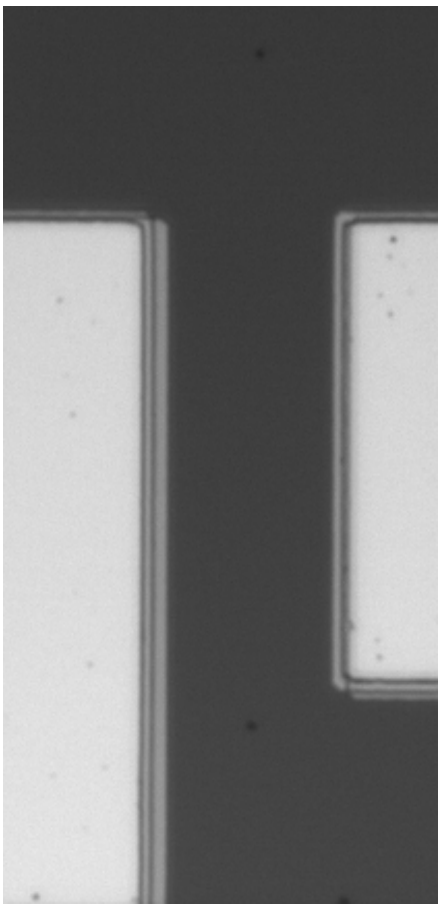
[9] *NI Vision: NI Vision Builder for Automated Inspection Development Toolkit Tutorial* [online]. USA: National Instruments, 2018, 81 s. [cit. 2019-08-31]. Dostupné z: http://www.ni.com/pdf/manuals/371424r.pdf

Příloha A
Doplňující snímky

A.1 Příklady pozorovaných hran

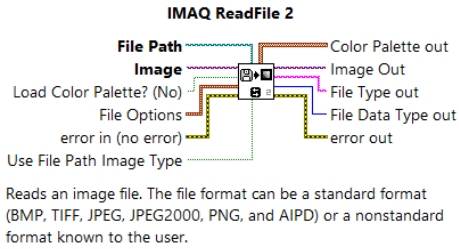


Obrázek A.1: Detail vad na hranách zrcátka vyniklých jeho opracováním.

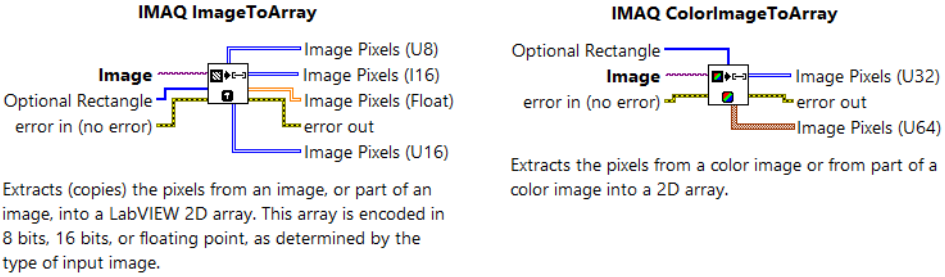


Obrázek A.2: Detail vícenásobné hrany vzniklé z důvodu nedokonalého dosednutí masky.

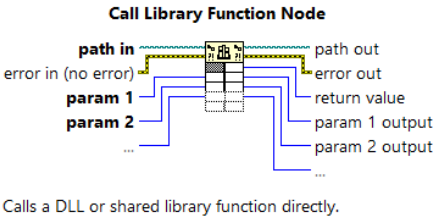
A.2 Použité funkční bloky v LabVIEW



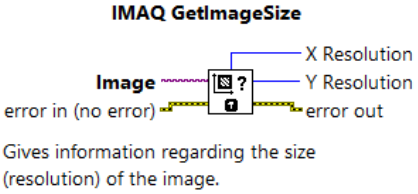
Obrázek A.3: Funkční blok na načtení uloženého obrázků z paměti. Převzato z nápovědy k LabVIEW.



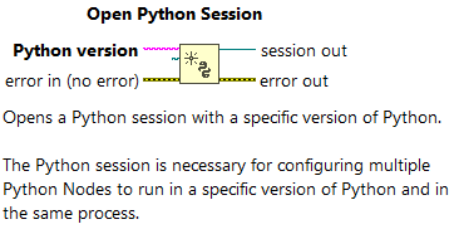
Obrázek A.4: Funkční bloky LabVIEW použité na převod obrázku formátu IMAQ Image.ctl na pole prvků. Převzato z nápovědy NI LabVIEW.



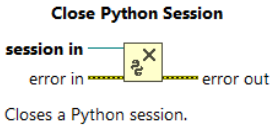
Obrázek A.6: Funkční blok volající DLL. Převzato z nápovědy NI LabVIEW.



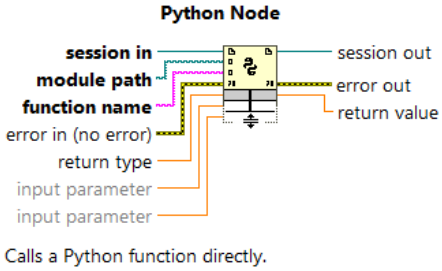
Obrázek A.7: Funkční blok pro zjištění rozměrů snímku. Převzato z nápovědy NI LabVIEW.



(a) : Blok pro spuštění procesu Python.



(b) : Blok pro ukončení procesu Python.



(c) : Blok pro volání skriptu v jazyce Python.

Obrázek A.5: Použité funkční bloky pro komunikaci s externími skripty v jazyce Python. Převzato z nápovědy NI LabVIEW.

Příloha B

Obsah přiloženého CD

```
/
├── README.txt
├── zigajannBP.pdf
└── VBAI
    ├── python32.vbai
    ├── python32.vi
    ├── python.py
    └── test_image.tif
```

Soubor zigajannBP.pdf obsahuje tuto bakalářskou práci ve formátu PDF. Ve složce VBAI se nachází soubory pro NI Vision Builder. Soubor python32.vbai slouží k otestování připojení externího skriptu v jazyce Python 2.7 (32 bitové verze) do NI Vision Builder. Do něj je připojen soubor python32.VI, který volá skript ze souboru python.py. Soubor test_image.tif obsahuje ukázkový snímek pro inspekci vytvořenou v programu VBAI.