VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÁ KONTROLA KVALITY VÝROBKU Z OBRAZU

AUTOMATIC INDUSTRIAL QUALITY CONTROL FROM IMAGE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **MARTIN KRUTÁK**
AUTHOR

VEDOUCÍ PRÁCE **prof. Dr. Ing. PAVEL ZEMČÍK**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Kruták Martin, Bc.**
Program: Informační technologie Obor: Počítačová grafika a multimédia
Název: **Automatická kontrola kvality výrobku z obrazu**
Automatic Industrial Quality Control from Image
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte literaturu a dostupná řešení pro detekci a zjišťování rozměrů objektů a vad povrchu objektů, to vše se zaměřením na kontrolu výrobků.
2. Navrhněte postup, s využitím metod umělé inteligence nebo i tradičních metod, pro zjištění/ověření rozměrů a detekci povrchových vad, například poškrábání.
3. Navrhněte způsob implementace takových postupů a zhodnoťte, jaké budou mít vlastnosti a jak mohou vyhovět v praxi.
4. Implementujte navržené postupy a demonstrujte funkčnost řešení.
5. Diskutujte dosažené vlastnosti a možnosti pokračování práce.

Literatura:

- Dle pokynů veducího

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3 zadání

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Zemčík Pavel, prof. Dr. Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 15. ledna 2019

Abstrakt

Cílem této práce je vytvořit postupy pro celkovou, automatickou a bezdotykovou kontrolu kvality výrobku (diabolky). Problém kontroly je rozdělen na dvě části. První část je přesné měření rozměrů diabolky – její délka a průměr hlavičky tak, aby toto měření bylo přesné a dostatečně rychlé. Takového měření je dosaženo pomocí algoritmů pracujících se sub-pixelovou přesností pomocí aproximace gradientů polynomy. V druhé části je kontrola vad diabolky jako jsou podélné rýhy či vrypy v sukýnce (smajlíci), a to za pomocí konvolučních neuronových sítí. Výsledkem jsou měřící moduly pracující s přesností do 0,025 mm v případě měření délky a do 0,01 mm při měření hlavičky. V detekci vad diabolek, vykazuje neuronová síť velmi vysokou úspěšnosti klasifikace. Přínosem této práce je prezentace inovativních přístupů v automatické kontrole kvality výrobků za použití neuronových sítí, demonstrace jejich použití a využití v reálné výrobě.

Abstract

The goal of this thesis is to create overall, automatic and non-contact quality control of a pellet. The issue is divided into two separate parts. The first part deals with precise dimensional measuring of pellet – its length and head diameter so that it is precise and reasonably fast. Precise measuring is achieved with help of algorithms which achieve the sub-pixel precision by polynomial approximation of the edges extracted from the image gradients. The second part deals with the defects of a pellet. Detecting defects like longitudinal furrows or skirt cuts is achieved with convolutional neural networks. The measurement modules work with the resulting precision up to 0.025 mm in case of length measuring and up to 0.01 mm in case of head diameter measuring. In case of defect detections, neural network shows very high classification success rate. The contribution of this thesis is a presentation of innovative approaches in automatic quality control of pellets with use of neural networks and a demonstration of its usage in real manufacturing process.

Klíčová slova

Konvoluční neuronové sítě, měření rozměrů výrobku, povrchová detekce defektů, počítačové vidění, strojové učení.

Keywords

Convolutional neural networks, dimensional measurement, surface defects detection, computer vision, machine learning.

Citace

KRUTÁK, Martin. *Automatická kontrola kvality výrobku z obrazu.* Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Dr. Ing. Pavel Zemčík

Automatická kontrola kvality výrobku z obrazu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana prof. Dr. Ing. Pavla Zemčíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Kruták
20. května 2019

Poděkování

Děkuji vedoucímu prof. Dr. Ing. Pavlu Zemčíkovi za vedení a rady během celé tvorby této práce. Děkuji dále společnosti Kinali s.r.o za poskytnutí prostor a prostředků pro vznik a vývoj této práce.

Obsah

Kapitola 1

Úvod

Velkou potřebou v mnoha výrobních odvětvích je přesnost a kvalita. Výrobky musí odpovídat určitým tolerancím v rozměrech a také různým normám, které jsou na výrobek kladeny. Takové normy ve většině případů kladou důraz na kvalitu výrobku, jeho odolnost, popř. jiné vlastnosti jako pružnost či bezpečnost. Kvůli nutnosti naplnění takovýchto norem, je třeba provádět kontroly výrobků a výrobní linky jako takové, aby se zaručila požadovaná výsledná kvalita za rozumných nákladů. Snahou předložené diplomové práce je přispět právě v těchto oblastech.

Bezdotykové měření a inspekce je s přicházejícím fenoménem „Průmyslu 4.0“ jedno z řešených témat. Možnost měřit jakékoli součástky v různých etapách výroby je velice cenná. Ve chvíli, kdy je kladen požadavek na to, aby se snížily náklady na výrobu a zvýšila produkce, je kladen vyšší důraz na důslednou kontrolu každého kusu výrobku, která by pomohla k velice rychlé identifikaci problémů v některé části výroby (deformace formy, chyba lisování, nedodržení postupu, apod.), která se ruku v ruce nese s okamžitou identifikací chyby a možného řešení závady, bez předchozí produkce několika desítek nebo dokonce stovek zmetků. V této oblasti se tedy jeví stále více jako výhodné využít moderních technologií k provádění takových úkonů. Přesněji pomocí algoritmů pro detekci vad a měření jen z obrazu kamery vyhodnotit kvalitu jednotlivých výrobků.

V mnoha provozech ještě dnes, stejně tak jako v dobách minulých, nalezneme postupy, které provádějí průběžnou kontrolu. Typicky to vypadá tak, že kontrolor jednou za čas (nebo za X výrobků) vezme jednu součástku a ručně či na přístroji (v závislosti na požadované přesnosti a velikosti výrobku) přeměří a vizuálně zkontroluje daný výrobek. Takový postup vede k chybám, kdy z kontrolního bodu vyjdou výrobky, na kterých vznikla vada třeba těsně po kontrole a přidá starosti obsluze dále na lince. Jiným přístupem je kontrola každého výrobku nebo sady výrobků skupinou pracovníků. V rámci urychlení výroby však v takových případech dochází k vyhazování i kusů, které by jinak prošly jako kusy správné. Jako příklad lze uvést linku na kontrolu dřevěných latěk, které chodí ve svazcích. Kvůli rychlosti výroby však daná obsluha optimalizuje svůj výkon tím, že zahlídne-li pár kusů vadných v daném svazku, vyhodí jej celý. Toto vede ke zbytečnému zvýšení objemu výhozu. Tento přístup má i další nevýhodu, a tou je nejednotnost mezi kontrolujícími. Může se totiž stát, že to co jeden kontrolor označí za vadný kus, druhý označí za kus správný. Dochází tak k nekonzistenci výroby, která se mění dle složení kontrolorů v dané směně.

Cílem této práce je vytvořit postupy pro bezdotykovou kontrolu diabolek. Tato kontrola spočívá ve dvou krocích. Prvním je přesné změření průměru hlavičky a také její délky. V druhém kroku je to detekce vad diabolek, mezi které patří např. podélné rýhy či rýhy v sukni způsobené vadnou formou.

Práce je dále koncipována následovně. Kapitola 2 rozebírá přístupy k měření, aplikace v oboru a matematické základy pro měření. V kapitole 3 jsou zmíněny teoretické základy neuronových sítí a jejich využití v oblasti detekce defektů. Základní definice obrazu a knihoven pro jejich zpracování je v kapitole 4. Kapitola 5 obsahuje zhodnocení současného stavu a návrh systému. Kapitola 6 shrnuje implementační detaily řešení spolu s vizualizacemi. Vyhodnocení implementovaných postupů je poté v kapitole 7. Závěry jsou shrnuty v kapitole 8.

Kapitola 2

Typy měření rozměrů výrobku a řešené aplikace v oboru

Měření[1] je v průmyslu jedna z nejvíce vyžadovaných činností. Lze ji provádět v mnoha formách. Od hrubých měření až po relativně přesná měření. V ideálních podmínkách lze měřit s velmi vysokou přesností.

Měření se také postupně přesunuje do oblasti informačních technologií, a tak stroje a mechanická měřidla nahrazují počítače, které pomocí speciálně vyvinutých algoritmů dokáží z obrazu kamery mnohonásobně rychleji změřit rozměry výrobku – délky, rádiusy, hloubky – mnohem rychleji a přesněji, než člověk fyzicky za pomoci mechanického měřidla. Měření za pomocí počítačového vidění navíc nabízí i vysokou přesnost, neboť při měření např. rozměrů dřevěných součástí či olověných výrobků je u mechanických měřidel problém tlaku, který je vyvinut na povrch výrobku, které může ovlivnit samotné měření až v desetinách milimetrů.

Vzhledem k rozsahu práce, jsou zde uvedeny jen skutečnosti, které jsou podstatné k práci samotné. Cílem této kapitoly není podat úplný přehled dané oblasti, jen její reprezentativní část.

2.1 Typy měření rozměrů

Z hlediska přístupu k měření rozměrů jako takovému lze rozlišit dva druhy měření – kontaktní a bezkontaktní. Oba přístupy pracují na určitém způsobu rekonstrukce povrchu či objektu a následného změření. Výhodou bezkontaktního měření je navíc fakt, že tyto systémy lze nasadit v případech, kdy zkoumaný objekt je natolik křehký, že měření, které by vyžadovalo kontakt s povrchem, by daný objekt zdeformovalo. Kontaktní měření je zde zmíněno jen pro úplnost. Hlavním záměrem v této části je představit bezkontaktní měření, přesněji, měření skrze industriální kamerové systémy.

Kontaktní měření

Význačnou vlastností systémů dotykového měření je nutný fyzický kontakt s měřenou součástí. Takový přístup se však ne vždy musí hodit. V případě potřeby vysoké rychlosti měření, dotykové systémy nemusejí stíhat. Dalším problémem dotykových systémů je fakt, že jsou

[1] Měření zde není myšleno přesně ve smyslu zákona č. 505/1990 Sb., Zákon o metrologii a jeho novely zákona č. 119/2000 Sb. a č. 85/2015 Sb. Pojem měření je použit čistě pro příhodnost daného termínu k vlastnostem řešených v této kapitole.

závislé na tvrdosti (křehkosti) materiálu, který měří. Nehodí se tedy na všechny povrchy. Výhodou je jejich jednoduchost.

Dotykové snímací systémy lze dělit do dvou kategorií [54]: systémy spínacího typu a systémy snímacího typu.

Systémy spínacího typu fungují tak, že při dotyku měřidla s povrchem měřeného objektu, se toto měřidlo zastaví a dochází k odečtení souřadnic měřícího systému stroje. Takový spínací systém pak může být elektromechanického nebo piezoelektrického typu [54].

Samotná spínací sonda se skládá z těla, modulu a dotyku. Přichycení modulu k tělu je provedeno magneticky, což umožňuje jednoduchou výměnu modulu v případě defektu třeba v důsledku nárazu. Taková sonda poté provádí odečet jen diskrétních hodnot. Jedná se tedy o kontaktní diskrétní snímání [54].

Elektromechanická sonda, má uvnitř elektrický obvod obsahující tři kontakty s kruhovou roztečí 120°. Ve chvíli dotyku s objektem, dojde k rozpojení elektrického obvodu a tím k odečtu hodnoty. Pro přesnější měření jsou elektrické kontakty kulového tvaru. Nevýhodou takového měření je zpozdění, které vzniká mezi okamžikem dotyku a rozpojením kontaktu, čímž vzniká určitá chyba měření. Této chybě se lze vyhnout použitím piezoelektrických snímačů [54].

Piezoelektrické sondy (viz Obrázek 2.1) se dle způsobu znamení dělí na dva druhy. Při indikaci pomocí piezokrystalů se elektrické signály vytvářejí při jejich deformaci. Druhý typ indikace je elektromechanický, který je popsán výše [54].

Silná stránka piezoelektrických senzorů je i jejich slabinou. Tyto senzory jsou tak citlivé, že mohou vydat signál i při záchvěvech stroje. Je tedy nutné informace z těchto snímačů ověřovat [54].

Systém snímacího typu (skenující sonda) „*se skládá z těla sondy, které je v pouzdře snímací hlavy a je uložené ve dvojitém nebo trojitém paralelogramu*“ [54].

Skenující sonda, na rozdíl od systémů spínacího typu, je v konstantním dotyku s povrchem. Tento konstantní dotyk je zajištěn pomocí pohonů os, které svou práci započnou v prvotní fázi dotyku měřidla s povrchem. Měření poté může probíhat ve dvou režimech. V případě statickém režimu dosáhneme tzv. diskrétního snímání. V režimu dynamickém se jedná o spojité snímání.

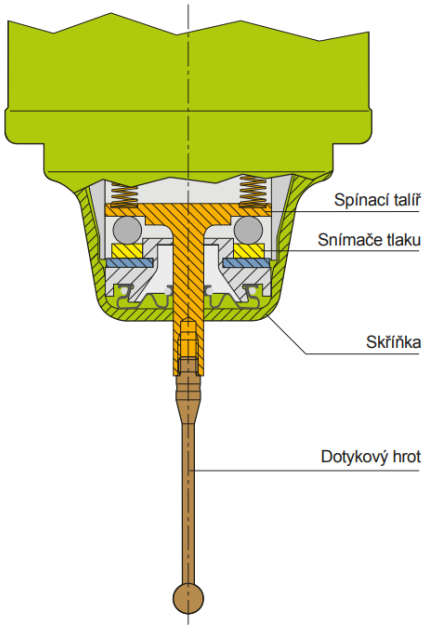
Bezkontaktní měření

Bezkontaktní měřící metody se dají dělit do čtyřech hlavních kategorií – optické, ultrazvukové, pneumatické a elektrické. Z nichž větší vývoj je zaměřen na první dvě [53]. Samostatnou skupinu tvoří měřící systémy založené na počítačovém vidění.

Optické metody jsou založené na bezkontaktním měření za použití světla, které se odrazí od povrchu objektu a je zachyceno CCD (Charge-coupled device) kamerou. Některé z optických metod jsou *světelného ostření, světlo-měřící a světelné* [53]. Existují ovšem i další metody.

Jedna z technik založených na *ostření světelných paprsků* pracuje na principu laseru a Dopplerova efektu. Tato technika pracuje na způsobu zachycení světla, které se tříští od povrchu rotujícího se objektu. Posuvem ve frekvenci odražené světla lze měřit kruhovou frekvenci. Obvod takového objektu je pak měřen na základě obvodové rychlosti a času, získané z úhlové rotace objektu [53].

[2] Převzato z [24].



Obrázek 2.1: Princip piezoelektrické sondy používané pro dotyková měření.[2]

Rozlišení takové techniky je závislé na rozmezích interferenčních proužků, ovšem i tak je přesnost ±20 µm ve vzdálenosti 80–100 mm, která je navíc závislá na vibracích samotného nástroje [53].

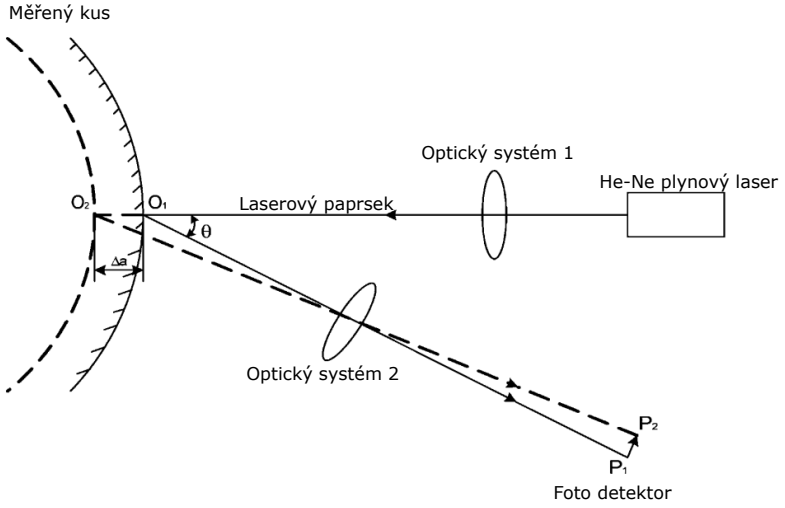
Světlo-měřící techniky využil např. Quibo (v [53]), který navrhl metodu měření objektů velkých rozměrů za potlačení Abbeho efektu (úhlová chyba se zvyšuje v závislosti na vzdálenosti od zdroje) s využitím heterodynového laserového interferometru. Měření probíhá tak, že jeden paprsek je rozdělen na dva na sebe kolmé paprsky. První z nich je použit jako referenční a druhý je použit jako měřící. Ve chvíli, kdy oba paprsky projdou fotodiodami umístěnými na stranách měřeného objektu, je změřen průměr objektu [53].

Čistě *světelné metody* měření se často skládají z laseru, optického systému a fotocitlivých zařízení (ukázka viz Obrázek 2.2). Měřící systém je prvně zkalibrován tak, aby ve výchozím stavu byla známá hodnota rozměru (v tomto případě průměru). Paprsek odražený z laseru, který míří kolmo na povrch objektu skrze optický systém, se odráží v místě O_1 do záznamového zařízení na pozici P_1. Ve chvíli, kdy dojde ke změně průměru měřeného objektu, dojde také k vychýlení paprsku. K odrazu dochází v místě O_2. Záznamové zařízení je třeba přesunout tak, aby doražený paprsek stále dopadal přímo do zařízení (pozice P_2). Tento posuv je vyjádřen v hladinách elektrického signálu, na základě kterého je změřen průměr výrobku jako odchylka od zkalibrované hodnoty [53].

Ultrazvukové metody využívají techniky šíření vln, které jsou frekvenčně vně spektra slyšitelného člověkem. Používají pulsně-akustickou techniku pro generování zvukových vln [53].

Pneumatické metody pracují na principu měření změny tlaku vzduchu mezi měřící hlavou a objektem. Tuto změnu poté převádí na elektrický signál. Jednou z výhod takového přístupu je, že konstantní proud vzduchu, který se pro měření používá, odstraňuje taktéž

[3] Převzato z [53].



Obrázek 2.2: Ukázka systému měřícího za pomocí laserového paprsku.[3]

tekutiny, které leží na povrchu měřeného objektu (např. mazadla použitého při řezání nebo železných pilin) [53].

Elektrické metody využívají pro měření odpor či vířivý proud. Tyto metody mají ovšem velkou nevýhodu, a to že měřený objekt musí být z podstaty vodivý. Jedním z možností, jak tohoto využít při měření je systém měřící průměr obrobku skrze servo jednotky. Každá taková servo jednotka se skládá ze senzoru vzdálenosti, krokového motoru a senzoru posuvu pro posuvný vozík [53].

Měření jako takové pak probíhá tak, že posuvný vozík se snaží zůstat v měřícím rozsahu senzorů. V případě, že dojde ke změně průměru měřeného kusu, dostane se obráběný kus z dosahu senzoru vzdálenosti a je nutné posunout vozík. Tento posuv zaznamená senzor posuvu vozíku a lze pak vypočítat změnu v průměru obrobku [53].

Měřící systémy založené na počítačovém zpracování obrazu mají výhodu rychlosti a přesnosti měření. Proto je lze nasazovat v případech výrobních linek vysokých taktů, kde je třeba změřit každý výsledný produkt. Určitou nevýhodou je fakt, že tyto systémy se skládají minimálně ze dvou částí a tou je kamerový systém, který zaznamená objekt ve vysokém rozlišení a samotný software, který detekuje objekt, vyextrahuje jej z obrazu a provede změření.

Kamery pro tento systém lze využít dvojího typu – kamery s čipem CCD (Charge-coupled device) a nebo čipem CMOS (Complementary Metal–Oxide–Semiconductor). Liší se svými vlastnostmi – zejména pak způsobem zachycení obrazu a specifikacích.

CCD kamery zaznamenávají obraz tak, že světlo dopadající na čip vytváří elektrický náboj, který je v tomto čipu uložen. Čip je rozdělen vodorovnými a svislými elektrodami, které vytváří na čipu mřížku, dělící jej na buňky. Každá jedna buňka reprezentuje pixel. Délka expozice, tedy času, po který je čip vystaven světlu, ovlivňuje množství náboje v něm uložené [43].

Ve chvíli, kdy je potřeba informaci z čipu vyčíst, jsou na vodorovné a svislé elektrody přivedena různá napětí a tím se náboje přelévají z jedné buňky do druhé směrem k výstupnímu zesilovači, kde je jejich náboj převeden na napěťovou úroveň [43].

Výhodou CCD čipů je nízký šum. Nevýhodou je jejich vyčítání po řádcích, problémy se saturací a vyšší spotřeba energie v porovnání s CMOS technologii [43].

CMOS kamery se od těch s čipem CCD liší způsobem záznamu a vyčítáním. Na čipu CMOS má každá buňka svůj obvod, který se stará o vyčítání hodnot z této buňky. Nastává tedy situace, že většinu čipu zabírají tyto obvody. Aby paprsky světla dopadaly jen na světlocitlivé části, je nad celým čipem sada čoček, které pomáhají soustředit paprsky přímo na danou buňku. Vyčítání z čipu CMOS pak, díky individuálním obvodům pro každou buňku, probíhá najednou, což urychluje vyčítání jako takové [51].

Nevýhody CMOS čipů jsou jejich vyšší šum. Výhody oproti CCD čipů jsou levnější výroba a rychlost vyčítání [51].

2.2 Analytické přístupy k měřícím úlohám

V následující části jsou zmíněny přístupy k přesnému měření rozměrů výrobku. Zaměření je zde jen na práce v oboru, kde probíhá bezdotykové měření s přesností pod pixel.

V případě analytických metod lze postupovat například následovně [40]:

1. *Pořízení obrazu* – pořízení snímku kamerou.
2. *Zpracování obrazu* – odfiltrování šumu, pozadí a předzpracování pro následující krok (popř. odstranění soudkovitosti a jiných deformací obrazu).
3. *Extrakce příznaků* – dochází k extrakci známých příznaků pro danou aplikaci. Tyto příznaky se extrahují tak, aby se vzájemně nepřekrývaly, což umožňuje lepší klasifikaci. Pro měřící úlohy je to často detekce hran. Extrahování hran se v aplikacích, které řeší přesně měření, neobejde bez sub-pixelové detekce hrany. Tohoto je většinou dosaženo proložením gradientu (okolí hrany) polynomem, pasování či pomocí momentů [13].
4. *Vyhodnocení* – kombinací vyextrahovaných příznaků dochází k vyhodnocení, zda-li v dané úloze objekt odpovídá zadaným tolerancím, popř. modelům.

Peng a kol. [45] navrhli systém pro měření průměru těsnících gumiček. Vstupem jejich systému je obraz gumového těsnění, který je zespodu podsvětlený. Samotné měření se skládá ze tří kroků – nalezení hrany s pixelovou přesností, dopřesnění hrany se subpixelovou přesností a proložení hrany kruhem.

Pro pixelové nalezení hrany autoři porovnali několik přístupů. Roberts, Sobel, Prewitt, Laplacian-Gauss, Canny a detekce hrany pomocí morfologických operací. Všechny přístupy, až na morfologii, zanechaly nějaké redundantní hrany na povrchu gumiček, které vznikly v důsledku odlesků. V konečném důsledku výzkumníci detekovali hranu pomocí morfologických operací, které podávají v daném případě nejlepší výsledky [45].

V dalším kroku dochází k dopřesnění hrany na subpixelovou přesnost. Výzkumníci navrhli metodu, která projde všechny body pixelově přesné hrany a v každém z nich hledá v okně velikosti 7×7 pixelů směr gradientu a nejvyšší bod. A to tak, že v hlavním směru a ve směrech natočených o 45° od hlavního směru, se sečtou gradienty bodů. Směr s nejvyšší hodnotou gradientu je zvolen jako výsledný. Všemi sedmi body ve výsledném směru se poté prokládá kubická křivkou. Nulový bod druhé derivace kubické křivky pak udává výslednou pozici gradientu s přesností pod pixel [45].

Ve chvíli, kdy jsou takto proloženy všechny body, prvně pixelově přesné hrany, přichází na řadu pasování kružnice. To je provedeno pomocí metody nejmenších čtverců. Všechny body však nejsou prokládány najednou. Body kruhu jsou rozděleny na osminy a jednotlivé

segmenty jsou proloženy zvlášť. Výsledný střed a rádius se počítá akumulací jednotlivých osmin. Důvody volby dělení na osminy autoři neudávají. [45].

Z předešlé práce je vidět, že zásadním krokem v měřících algoritmech je přesné, tedy sub-pixelové detekování hrany. Přístupy v sub-pixelové detekci hrany se dají rozdělit do tří kategorií – interpolační metody, metody založené na momentech a metody pracující s nejmenší čtvercovou chybou.

Interpolační metody se snaží dosáhnout subpixelové přesnosti proložením hodnot daných pixelů (diskrétních jednotek) nebo jejich derivací. Tuto metodu vyžili již zmíněný Peng a kol. [45], ale také Steger [49], který prokládat hodnoty gradientního obrazu polynomem druhého řádu pro přesnou detekci hrany. Hermosilla a kol. [25] navrhli řešení, kde použili Canny detektor na prvotní odhad pozice hrany a poté Hermitovu interpolaci pro její přesnější dohledání. Gao a kol. [19] zase použili pro přesnou detekci hrany fraktální derivaci Newtonovy interpolace.

Metody založené na momentech používají momenty obrazu pro detekci hrany. Tyto metody nejsou iterativní a obejdou se bez interpolací. Machuca a Gilbert [39] byli prvními, kteří navrhli hledat hranu pomocí momentů. Autoři nejprve integrují oblast, ve které se hrana nachází a poté na základě momentů dané oblasti, definované vlastnostmi vektoru z daného pixelu a jeho okolí do jeho těžiště, určí pravděpodobnost příslušnosti daného bodu hraně. O poznání vyššího úspěchu se těší jejich následnící Tabatabai a Mitchell [50], kteří detekují subpixelovou pozici hrany pomocí pasování tří momentů intenzity do modelu ideální hrany. Bin a kol. [4] použili pro hledání subpixelové hrany ortogonální Fourier-Mellin momenty. Momentů počítají celkem 7 s využitím kruhové masky.

Metody založené na nejmenší čtvercové chybě (z angl. least-squared-error-based) se snaží najít subpixelovou pozici hrany pasováním hodnot do předpokládaného modelu hrany. Ye a kol. [60] navrhli metodu pro detekci hrany s vysokou přesností pomocí Gaussova modelu. Nalwa a Binford [44] nabízí metodu založenou na nejmenší čtvercové chybě, která první odhadne orientaci hrany napasováním roviny. Hyperbolická funkce je poté napasována ve směru hrany.

2.3 Matematické základy měřících metod

Pro měřící úlohy je důležité prokládání pro subpixelové nalezení vrcholu gradientu – polynomem, křivkou, či celým útvarem – kružnicí, elipsou. Tato část neposkytuje kompletní výčet, ale jen zástupce z každé kategorie.

Licování kružnice metodou nejmenších čtverců

Jeden způsob pasování kružnice metodou nejmenších čtverců publikovala Bullock [7]. Jádro metody zní následovně. Mějme střed kruhu (u_c, v_c) a rádius R. Metoda spočívá v minimalizaci funkce $S = \sum_i (g(u_i, v_i))^2$, kde $g(u, v) = (u - u_c)^2 + (v - v_c)^2 - \alpha$, kde $\alpha = R^2$. K docílení takového výsledku je třeba, aby $\partial S / \partial \alpha = 0$, čehož lze dosáhnout tehdy a jen tehdy, když

$$\sum_i g(u_i, v_i) = 0 \tag{2.1}$$

V návaznosti na rovnici 2.1 je třeba zjistit, kdy $\partial S/\partial u_c = 0$ resp. pro v_c, $\partial S/\partial v_c = 0$, a to lze tehdy a jen tehdy, když

$$\sum_i u_i g(u_i, v_i) = 0 \tag{2.2}$$

respektive

$$\sum_i v_i g(u_i, v_i) = 0 \tag{2.3}$$

Definujme $S_u = \sum_i u_i$, $S_{uu} = \sum_i u_i^2$, apod. Ze znalosti $S_u = 0$ resp. $S_v = 0$, lze rovnici 2.2 rozložit na tvar

$$u_c S_{uu} + v_c S_{uv} = \frac{1}{2}(S_{uuu} + S_{uvv}) \tag{2.4}$$

respektive

$$u_c S_{uv} + v_c S_{vv} = \frac{1}{2}(S_{vvv} + S_{vuu}) \tag{2.5}$$

Vyřešením soustavy rovnic 2.4 a 2.5 získáme (u_c, v_c), což je střed kružnice v souřadném systému posunutém o střední hodnotu všech vstupních bodů v x a y souřadnicích. Pro získání souřadnic v původním souřadném systému, je nutné tyto střední hodnoty nakonec k získanému středu přičíst.

Výsledný rádius lze vypočítat roznásobením rovnice 2.1 a tedy

$$\alpha = u_c^2 + v_c^2 \frac{S_{uu} + S_{vv}}{N} \tag{2.6}$$

kde N je počet vstupních bodů. Konečně $R = \sqrt{\alpha}$.

Pasování elipsy

Algoritmus pro pasování elipsy navrhl Fitzgibbon a kol. [17]. Výhodou tohoto algoritmu je, že vrátí rovnici elipsy i pro špatně definovaná data a je navržený tak, aby byl i výpočetně rychlý a jednoduše implementovatelný.

Autoři derivují problém pasování elipsy z problému pasování kužele. Mějme tedy obecný kužel, který lze reprezentovat pomocí polynomu druhého řádu

$$F(\boldsymbol{a}, \boldsymbol{x}) = \boldsymbol{a} \cdot \boldsymbol{x} = ax^2 + bxy + cy^2 + dx + ey + f = 0, \tag{2.7}$$

kde $\boldsymbol{a} = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix}^T$ a $\boldsymbol{x} = \begin{bmatrix} x^2 & xy & y^2 & x & y & 1 \end{bmatrix}^T$. Pasování obecného kuželu může dále být provedeno minimalizováním součtu čtvercových vzdáleností

$$\mathcal{D}_A(\boldsymbol{a}) = \sum_{i=1}^{N} F(\boldsymbol{x})^2 \tag{2.8}$$

křivky o N bodech $\boldsymbol{x}_i$.

V závislosti na rovnici 2.8 je výhodné, když vektor $\boldsymbol{a}$ bude omezen podmínkou tak, aby daný kuželovitý tvar byl nucen reprezentovat elipsu. Tato podmínka je známá, přesněji tedy, že diskriminant $(b^2 - 4ac)$ bude mít zápornou hodnotu. Tohoto lze dosáhnout vynucením

podmínky rovnosti $4ac - b^2 = 1$, kterou lze pomocí matice $\boldsymbol{a}^T\boldsymbol{C}\boldsymbol{a} = 1$ representovat jako

$$\boldsymbol{a}^T \begin{bmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{a} = 1 \tag{2.9}$$

Cílem je tedy minimalizovat $E = \left\|\boldsymbol{D}\boldsymbol{a}\right\|^2$ za podmínky $\boldsymbol{a}^T\boldsymbol{C}\boldsymbol{a} = 1$, kde matice $\boldsymbol{D}$ je $n \times 6$ matice $\begin{bmatrix}\boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n\end{bmatrix}^T$. Představením Lagrangeova násobitele vznikne soustavu rovnic

$$\begin{aligned} 2\boldsymbol{D}^T\boldsymbol{D}\boldsymbol{a} - 2\lambda\boldsymbol{C}\boldsymbol{a} &= 0 \\ \boldsymbol{a}^T\boldsymbol{C}\boldsymbol{a} &= 1 \end{aligned} \tag{2.10}$$

Tyto rovnice mohou vést až k šesti řešením. Z nich se volí takové, které má nejmenší zbytek $\hat{\boldsymbol{a}}_i^T\boldsymbol{D}^T\boldsymbol{D}\hat{\boldsymbol{a}}_i = \lambda_i$. Pro podrobnější popis i s důkazy viz [17].

Řešení soustavy rovnic pomocí QR dekompozice

QR dekompozice se využívá k řešení soustav lineárních rovnic a k výpočtům vlastních čísel matic. Její výhodou je, že nevyžaduje, aby matice byly čtvercové. [34]

Nechť A je matice $m \times n$ s lineárně nezávislými sloupci. QR dekompozice matice A je faktorizace $A = QR$, kde Q je $m \times m$ ortogonální matice a R je $m \times n$ je horní trojúhelníková matice. Tato dekompozice lze řešit třemi způsoby: [34]

1. Pomocí Householderových matic.
2. Pomocí Givensovy rotace.
3. Pomocí Gram–Schmidtovy ortogonalizace.

V této práci je využita dekompozice pomocí Givensovy rotace. Dále tedy bude přiblížena tato metoda.

QR dekompozice počítá $Q^TA = R$, což se dá také zapsat jako

$$\begin{bmatrix} \gamma & -\sigma \\ \sigma & \gamma \end{bmatrix}^T \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix},$$

kde $\gamma^2 + \sigma^2 = 1$. Matice Q^T se nazývá *Givensova rotace* [34].

Algoritmus výpočtu QR dekompozice pomocí Givensovy rotace je následující. Nechť $[c, s] = \boldsymbol{givens}(a, b)$ je funkce, která vypočte c a s tak, že

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix}^T \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}, \quad r = \sqrt{a^2 + b^2}.$$

Nechť $G(i, j, c, s)^T$ je matice Givensovy rotace, která orotuje $i$té a $j$té elementy vektoru $\mathbf{v}$ o úhel θ tak, že $\cos\theta = c$ a $\sin\theta = s$ tak, je-li $v_i = a$ a $v_j = b$ a $[c, s] = \boldsymbol{givens}(a, b)$, poté v novém vektoru $\mathbf{u} = G(i, j, c, s)^T\mathbf{v}, u_i = r = \sqrt{a^2 + b^2}$ a $u_j = 0$. Samotná QR faktorizace matice A velikosti $m \times n$ se vypočítá následovně. [34]

$Q = I$
$R = A$
for $j = 1 : n$ **do**
  for $i = m : -1 : j + 1$ **do**
    $[c, s] = \boldsymbol{givens}(r_{i-1,j}, r_{ij})$
    $R = G(i - 1, i, c, s)^T R$
    $Q = QG(i - 1, i, c, s)$
  end for
end for

Matice Q je akumulována sloupcovou rotací matice identity, neboť matice A je násobena maticí Q^T pro zredukování A na horní trojúhelníkovou matici. [34]

V této práci bude pomocí QR dekompozice metodou Givensovy rotace řešena soustava rovnic pro hledání polynomu druhého řádu – paraboly. Jednou z potřebných vlastností paraboly je znalost jejího vrcholu. Za předpokladu obecné rovnice paraboly $f(x) = ax^2 + bx + c$, lze pro tento výpočet využít následující vzorec:

$$V = \left[\frac{-b}{2a}; \frac{4ac - b^2}{4a}\right] \tag{2.11}$$

Znalost pozice vrcholu je důležitým krokem při hledání sub-pixelových hran v případě jejich prokládání parabolou. Lze se tak vyhnout integracím.

Kapitola 3

Přístupy k detekci defektů výrobků

Jednotlivé přístupy v automatické kontrole se velice liší v závislosti na daném problému, který řeší. Jiný přístup se zaujímá v případě kontroly kvality ovoce (velice populární téma) [37, 12, 5], diametrálně jiný v kontrole kvality kovu [47] a svůj zvláštní přístup si zaslouží i keramika [14]. Každá úloha kontroly kvality výrobku se liší v závislosti na specifických vlastnostech materiálu a vzhledu výrobku jako takového. Obecně lze vysledovat trend, kdy se kříží detekce defektů pomocí strojového učení s detekcí pomocí konvenčních metod (ručně implementovaná extrakce hran, významných bodů, apod.).

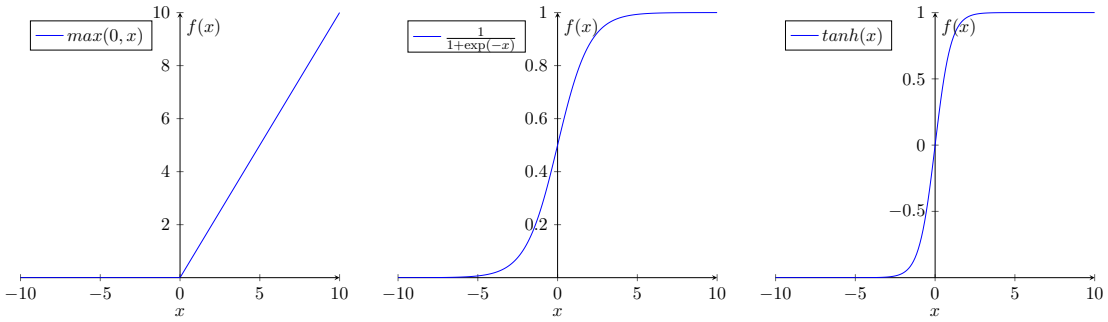
V následující kapitole je současný přehled obou přístupů. Jsou představeny práce v oblasti zabývající se detekcí defektů jak konvenčními metodami, tak za pomocí neuronových sítí. S přihlédnutím k rozsahu této práce, je cílem podat reprezentativní, nikoli vyčerpávající, přehled.

3.1 Stavba konvolučních neuronových sítí

Konvoluční neuronové sítě mají dané stavební bloky, ze kterých se tvoří. Každý následující blok logicky navazuje na předcházející a plní svůj účel [56, 57]. Vstupní obraz může být jak černobílý, tak barevný. Volba vstupu závisí na specifikách dané úlohy, kterou má výsledná síť řešit.

Konvoluce

Konvoluční vrstvy aplikují sadu filtrů na vstupní obraz. Takové filtry bývají malých dimenzí, ale aplikují se skrze celou hloubku obrazu (př. u barevného obrazu skrze všechny 3 kanály). Typické filtry mají velikosti 5×5 a 3×3. Ani větší filtry nejsou výjimkou, ovšem snižuje se tím rychlost celé neuronové sítě. Trendem je tedy používat místo rozměrných filtrů spíše sérii menších filtrů. Hojně se také využívá separability filtrů. Po aplikaci filtrů na celý obraz, vznikne 2D aktivační mapa, která mapuje odezvy filtru pro každou polohu v prostoru obrazu. Síť se takto naučí, které filtry se aktivují v případech, kdy se v daném místě zobrazí např. hrana či nějaká barva. Propojením těchto filtrů v různých vrstvách vznikne množina aktivačních map, které seřazené za sebou dávají výslednou hodnotu výstupu. Důležitým parametrem konvoluční vrstvy je také krok filtru se kterým je filtr aplikován. Určuje vzdálenost mezi jednotlivými aplikacemi filtrů. Častou volbou pro takový krok je 1. Pro hodnoty kroku vyšší jak 1 se konvoluční vrstva chová také jako pooling vrstva, čehož se často využívá.



Obrázek 3.1: Nejpoužívanější aktivační funkce zleva: ReLU, sigmoid a tanh.

Aktivační funkce

Jelikož reálný svět není lineární (v případě použití jen skalárního součinu by výstup ze sítě byl jen lineární), je třeba představit nelinearitu v síti. K tomuto účelu slouží (nelineární) aktivační funkce, které dovolují aproximovat jakkoli složité funkce. Nejrozšířenější aktivační funkcí je **ReLU** (Rectified Linear Units), která přidává nelineární transformaci k výstupu z konvoluční či plně propojené vrstvy. Tato aktivační funkce mapuje vstup na výstup v případě, že je vstup větší jak 0. V opačném případě (záporný vstup) je na výstupu jen 0 (viz Obrázek 3.1). Funkce je ve své podstatě primitivní $f(x) = max(0, x)$. Další z aktivačních funkcí je **Sigmoid**, která je taktéž nelineární, tedy je možné je umisťovat za sebe bez toho, aby jejich hodnota rostla nade všechny meze. Nevýhodou této funkce je, že na okrajích se dostává do situace, kdy má funkce velice nízkou odezvu. Jedná se o problém tzv. mizících gradientů [26]. V důsledku nízké odezvy je učení v takové fázi již velice pomalé. Poslední zmiňovanou funkcí je **Tanh**. Jedná se prakticky o funkci sigmoid s měřítkem ($tanh(x) = 2sigmoid(2x) - 1$). Její vlastnosti jsou lepší v tom směru, že nabízí silnější gradienty, ovšem taktéž trpí problémem s jejich mizením na okrajích funkce.

Pooling

Pooling vrstva slouží k nelineárnímu podvzorkování obrazu v obou rozměrech, což vede ke snížení rozměru na výstupu. Cílem této vrstvy je snížit množství parametrů sítě a výpočetní náročnost. Pooling vrstva se vkládá většinou mezi konvoluční vrstvy.

K provedení poolu lze využít taktéž několik strategií. Nejpoužívanější je max pooling, který na výstup pošle maximální hodnotu ze zkoumaného okolí. Dalšími méně používanými strategiemi je sum pooling (na výstupu je součet zkoumaného okolí vstupní příznakové mapy), mean pooling (průměrná hodnota vstupních hodnot) a global pooling (libovolný rozměr obrazu je podvzorkován na velikost 1×1).

V dnešní době se od pooling vrstev pomalu upouští, dají se totiž nahradit konvolučními vrstvami, které aplikují filtry s větším krokem. V budoucnosti se tak může stát, že se od pooling vrstev úplně upustí [29].

Plně propojená vrstva

Výstup z poslední konvoluční vrstvy je obvykle zploštěn do jedno-dimensionálního vektoru a připojen k jedné či více plně propojených vrstev. V této vrstvě je každý vstup propojen na výstup učící váhou. Počet výstupů této vrstvy se obvykle rovná počtu tříd, které má daná síť klasifikovat. Každý z těchto výstupů poté representuje pravděpodobnost příslušnosti

k jedné ze tříd. Toho dosahuje pomocí tzv. softmax funkce

$$p(a_i) = \frac{e^{a_i}}{\sum_{i=1}^{K} e^{a_i}}, \tag{3.1}$$

kde K je počet výstupů, a_i označuje *i-tý* prvek vektoru a $p(a_i)$ označuje predikovanou pravděpodobnost. I plně propojené vrstvy lze nahradit konvolučními vrstvami, neboť funkční vlastnosti obou vrstev jsou prakticky stejné. Obě provádí skalární součin [29].

3.2 Využití neuronových sítí v kontrole kvality

V situacích, kdy není jednoduché zcela přesně definovat posloupnost operací pro zpracování obrazu tak, aby vznikl v mezích robustní algoritmus na detekci určité vady, nabízí se využití neuronových sítí jako dobrá alternativa k analytickému přístupu. V některých případech jsou neuronové sítě ve vyhodnocení rychlejší, jak ty čistě analytické.

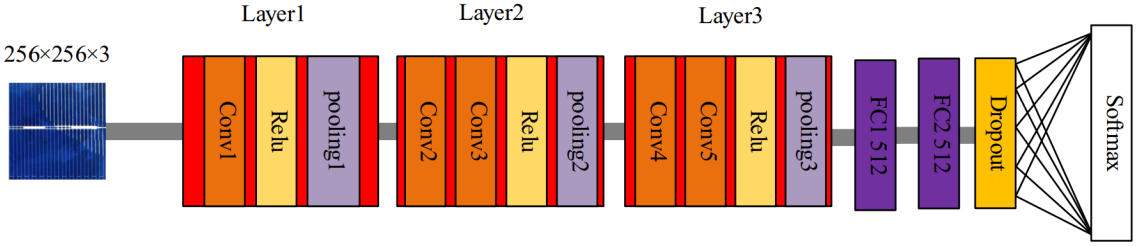
V této části jsou představeny některé přístupy detekce vad pomocí neuronových sítí pro různé objekty. Množství objektů a vad, které se dá pomocí neuronových sítí analyzovat (naučit a detekovat) je tak rozsáhlé a různé, že se lze v přístupech k problémům v oblasti jen inspirovat a vzít některé přístupy jako vstupní body pro řešení aplikovaného problému. Tato oblast je natolik mladá[1] a rychle se vyvíjející, že většina řešených problémů právě třeba pro kontrolu kvality povrchu výrobků je buďto neexistujících nebo zastaralá.

Chen a kol. [8] se zabývali detekcí defektů na solárních panelech. Jejich přístup spočívá v rozdělení obrazu samotného solárního článku na menší podčásti. Jeden obraz tak rozdělí na 49 částí, které poté manuálně roztřídili na části, které obsahují defekt a ty, které jsou bez defektu. Tento přístup zvyšuje rychlost učení při zanechání informace o daném defektu. Svou síť postavili na modelu Alexnet, kterou upravili, neboť defekty solárních článků mají velké rozdíly v různých spektrech. Vytvořili tak multi-spektrální konvoluční neuronovou síť, jejíž model lze vidět na Obrázku 3.2. Stejný model je použit i pro ostatní kanály. Obraz je tedy prvně na vstupu rozdělen na tři další dle barevných kanálů (RGB), jejichž výsledek je poté spojen posledními dvěma plně propojenými vrstvami (fully connected layers). Chen a kol. ve svém výzkumu dále zmiňují dopad volby faktorů jako je hloubka neuronové sítě a velikost jádra filtrů použitých při konvoluci. Při modelu s pěti vrstvami se přesnost sítě zvýšila o 2 % v porovnání se sítí se třemi vrstvami. Velikost jádra, na druhou stranu, však takový dopad nemá. Při zvýšení velikosti jádra totiž přesnost sítě vzrostla maximálně o 0,6 %.

Detekcí defektů na čipu LED diody se zabývali Lin a kol. [38]. Ve své síti, kterou nazvali LEDNet využívají techniku mapování dle aktivace třídy (z angl. Class activation mapping, dále jen CAM), která se vyznačuje lokalizací regionů specifických pro danou třídu. Podle tohoto přístupu se za poslední konvoluční vrstvu řadí další pooling vrstva s použitím strategie globálního průměru. Výsledná architektura konvoluční neuronové sítě pro kontrolu kvality čipu LED se skládá z šesti konvolučních vrstvev z nichž první, druhá a pátá je následována pooling vrstvou s max pooling strategií a za šestou vrstvou je pooling vrstva se strategií počítající průměr hodnot. CAM se poté vytváří v plně propojené vrstvě váženým součtem příznakových map. Výzkumníci vyhodnocovali svoji architekturu ve smyslu

[1] I přesto, že stochastický gradientní sestup (z angl. stochastic gradient descent), známý hlavně ze strojového učení, spatřil světlo světa již v roce 1952 [30], boom v oblasti strojového učení a umělé inteligence přichází až s nástupem hardwaru s masivní paralelizací výpočtů a frameworků, které nabízí rychlé sestavení celé sítě.

[3] Převzato z [8].



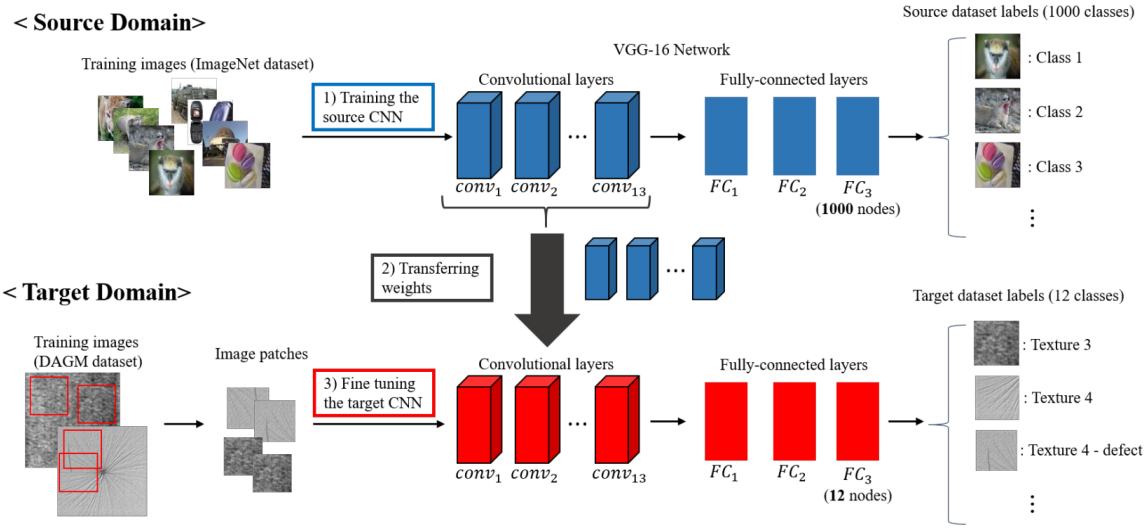
Obrázek 3.2: Model neuronové sítě použité v práci Chen a kol. [8] pro detekci defektů na solárním článku. Na obrázku je model pro jeden ze tří kanálů barevného obrázku. Pro ostatní je však, až na plně propojené vrstvy, stejný.[3]

nepřesnosti. Síť je nepřesná ve chvíli, kdy kategorie s nejvyšší jistotou je různá od správné kategorie. V tomto smyslu je udávaná nepřesnost sítě LEDNet 5,51 %. Slabinou hodnocení výzkumníků Lin a kol. je fakt, že sadu k vyhodnocení si výzkumníci vygenerovali sami (žádná oficiální sada na testování právě tohoto problému do té doby neexistovala) a tu samou sadu testovali oproti starší síti výzkumníků Kuo a kol. z roku 2014, aby dokázali svou vyšší úspěšnost (resp. nižší nepřesnost).

Výhody přeneseného učení (z angl. transfer learning) využili ve své práci Kim a kol. [31]. Přeneseného učení se dá podle nich využít i v situacích, kdy datová sada zdrojové sítě si není podobná s datovou sadou cílové sítě. V takovém případě je důležitým krokem doladění cílové sítě (v průběhu trénování v cílové síti se upravují i váhy v konvolučních vrstvách), které významně zvyšuje její přesnost. Kim a kol. představili postup (viz Obrázek 3.3), o kterém tvrdí, že se dá použít na jakýkoli problém v oblasti automatické optické kontroly. Jejich přístup spočívá ve využití existující architektury neuronové sítě – přesněji VGG-16, která se skládá ze 13 konvolučních vrstev a tří plně propojených vrstev. Po každých dvou až třech konvolučních vrstvách se nachází max pooling vrstva. Výzkumníci použili tuto síť již předtrénovanou na ImageNet datasetu, který obsahuje 1000 tříd. Takto předtrénovaná síť nabízí 7,5% chybovost pro validační sadu a 7,4% chybovost pro testovací sadu. Pro cílovou síť využili Kim a kol. taktéž síť VGG-16, ovšem plně propojené vrstvy upravili ke svému účelu tak, aby na svém výstupu měly 12 uzlů, což je počet tříd v datasetu pro průmyslovou optickou kontrolu, kterou vytvořila německá asociace pro rozpoznávání vzorů (DAGM). Cílovou síť nainicializovali dle vah ze zdrojové (naučené sítě) a plně propojené vrstvy nainicializovali náhodně. Učení pak probíhalo tak, že obrázky ze zdrojové sady rozřezali na překrývající se regiony o velikosti odpovídající vstupním velikostem sítě VGG-16 (224 × 224 pixelů). Výzkumníci poté vyhodnocovali přesnost pro tři scénáře: síť natrénovaná od úplného začátku jen na cílovém datasetu, cílová síť se zamraženými váhami v konvolučních vrstvách a síť s dolaďováním vah v konvolučních vrstvách. Výsledná síť s doladěním dosahuje téměř 100% přesnosti ve všech testovacích případech na rozdíl od dalších dvou sítí, jejichž přesnost se mezi testovacími případy velmi mění.

Faghih-Roohi a kol. [15] se ve svém výzkumu zabývali automatickou kontrolou defektů na kolejnicích. Jejich síť se nezakládá na žádné z populárních architektur (AlexNet, ResNet, GoogLeNet, ..). Pro porovnání rychlostí trénování a úspěšností výzkumníci navrhli celkem tři sítě, které se mezi sebou liší v parametrech – počet konvolučních vrstev, počet vrstev, počet příznakových map a počet uzlů plně propojených vrstev. Jejich konečné zjištění bylo, že i přesto, že všechny sítě (ve všech třech velikostech) nabízejí velice podobnou a solidní

[5] Převzato z [31].



Obrázek 3.3: Způsob využití přeneseného učení, které se skládá ze zdrojové sítě naučené na obecném datasetu, přenesení naučených vah do cílové sítě a doučení (doladění) na cílové datasetu.[5]

klasifikační úspěšnost (92 %), co se týče výkonnosti, je na tom nejlépe ta největší z nich. I přesto, že její trénování trvá nejdéle. Výsledná architektura té největší sítě se skládá ze třech konvolučních vrstev, z nichž každá je následována max pooling vrstvou. Na konci jsou poté zařazeny tři plně propojené vrstvy.

3.3 Analytické přístupy v detekci povrchových defektů

Analytické přístupy spočívají v kompletním vytvoření algoritmu detekce vad jen pomocí konvenčních metod. Tyto přístupy jsou tedy často náročnější a vyžadují tedy více času na vývoj.

Příznaky pro analýzu textur, které vedou k analýze defektů jsou obvykle extrahovány v prostorové nebo frekvenční doméně. Poté jsou použity různé rozlišovací metody pro rozlišení oblastí s defekty od pozadí na základě dříve extrahovaných příznaků.

Z rozlišovacích metod lze využít Bayesův teorém [32], binární lineární klasifikátory [27], neuronové sítě [55] či metody založené na vzdálenostech [9].

V prostorové doméně lze příznaky extrahovat z histogramu šedotónového obrazu nebo s využitím spolehlivějších šedotónových co-occurrence matic[6][35]. Mezi další způsoby, jak popsat textury jsou statistické metody (momenty obrazu) jako průměr, odchylka, míry asymetrie (v angl. skewness) a míra „špičkovitosti“ (v angl. kurtosis) extrahované z šedotónového histogramu [2].

Detekce defektů pomocí spektrální analýzy je více častá, než analýza v prostorové doméně. Jednou z výhod je fakt, že tyto metody jsou méně choulostivé na šum a změnu intenzity. Mezi takové přístupy lze řadit vlnkovou transformaci (VT), Fourierovu transformaci (FT), Gaborovu transformaci (GT) a klasické filtrování.

[6]Matice popisující, jak často se páry pixelů specifických hodnot intenzity a specifických vzdáleností objevují v obraze [48].

Fourierova transformace je nástroj, který slouží na dekompozici obrazu (signálu) na dvě složky – sinové komponenty a kosinové komponenty [16]. Výhodou FT je, že popisuje obraz jako celek a je schopna lépe popsat charakteristiky defektu za cenu ztráty lokální informace. Detekci defektů na textilu za pomocí analýzy Fourierovy transformace využili například Chan a Pang [35], kteří se zaměřili na centrální spektrum, nebo Chiu a kol. [10], kteří vytvořili metodu, která je invariantní vůči změně měřítka, rotaci či posunutí intenzit obrazu.

Vlnková transformace je způsob, jak dekomponovat signál na několik rozlišení. Její výhodou je fakt, že na rozdíl od FT jsou VT složeny z malých vlnek, které se mění ve frekvenci a délce trvání. Tato vlastnost dovoluje VT popsat lépe lokální informace v jakémkoli směru [41]. VT jsou využity často v detekci defektů na oděvních látkách [22, 58]. Yang a kol. [59] navrhli extraktor příznaků založený na adaptivních vlnkách s detektorem defektů pracujícím na základě Euklidovských vzdáleností. Se svým přístupem dosáhli přesnosti detekce 97,5 % v případě známých vzorků a 93,3 % v případě neznámých vzorků.

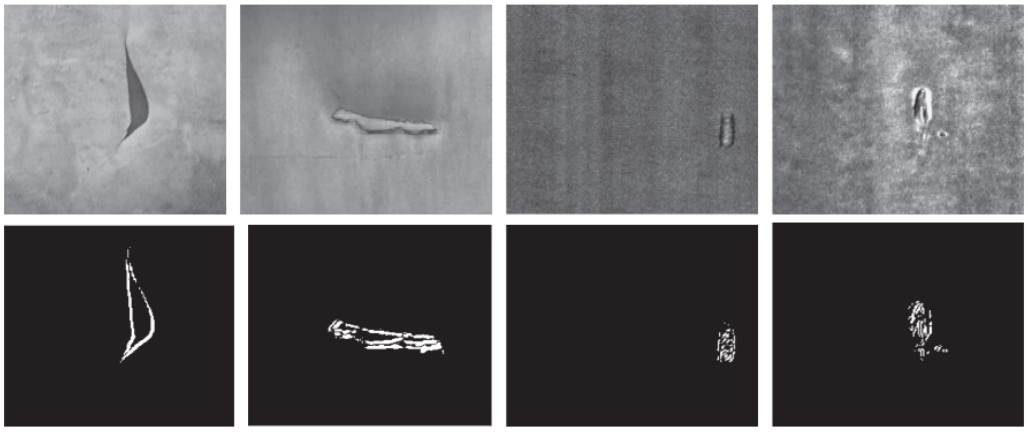
Gaborovy transformace jsou prakticky speciálním případem Fourierovy transformace. Gaborova transformace lokalizuje Fourierovu transformaci v určitém čase (okně) [18]. Kumar a Pang [33] navrhli využití Gaborových filtrů (pro 2D prostor) v detekci defektů obecně texturovaných povrchů. Své využití v detekcí defektů nalézá i Gaborova vlnka. Její aplikací se podařilo Tsai a kol. [52] detekovat defekty homogenních povrchů dřeva, smirkového papíru či tkanin. I když přesné vyhodnocení systému nebylo provedeno, dle autorů výsledný systém vyhodnocuje vzorky s vysokou shodou k rozdělení provedené člověkem.

Přístupy založené na filtrování jsou jednou z možností, jak efektivně detekovat defekty. Tyto přístupy mají své zastoupení spíše v detekcích defektů objektů náhodných textur. Monadjemi a Mirmehdi [42] využili ve své práci párování eigen filtru pro detekci anomálií v náhodných texturách. Prvně extrahovali eigen vektory z několika malých částí vzorku textury, která neobsahovala žádný defekt, a to pomocí analýzy hlavních komponent (z angl. Principal Component Analysis). Takto extrahované eigen vektory se nazývají eigen filtry. Testovaný obraz je poté rekonstruován pomocí podmnožiny vlastních eigen vektorů a podmnožiny natrénovaných eigen filtrů. Pokud je chyba takové rekonstrukce vyšší než předem zvolený práh, může se jednat o abnormalitu dané textury. Autoři udávají přesnost takového přístupu na 90 %.

Analýzu povrchu pomocí filtrování využili i Zhuang a kol. [63], kteří navrhli postup, jakým detekovat praskliny na solárních panelech. Jejich algoritmus pracuje s černobílým obrazem. Pořízený snímek je tedy prvně převeden na šedotónový. V druhém kroku následuje předzpracování obrazu. Kvůli rozptýlenosti hodnot v obraze je provedena ekvalizace histogramu a poté je obraz invertován a převeden na černobílý.

Vzhledem k charakteristikám prasklin, které se vyznačují ostrou změnou gradientu v místě praskliny, je v dalším kroku aplikován Laplaceův operátor. Kvůli přítomnosti šumu je nutné použít i Gaussův filtr na jeho částečné odstranění. Tyto operace lze na obraz aplikovat v jednom kroku. Je tedy vytvořeno Gauss-Laplaceovo jádro, po jehož aplikaci jsou v obraze patrné okraje prasklin. Pro odstranění malých bodů a za účelem propojení detekovaných regionů je následně aplikována morfologická operace uzavření (morfologická operace dilatace následovaná erozí).

Po přehození barev černé s bílou tak, aby pozadí bylo bílé, je provedena detekce kontur. Hledání kontur probíhá v osmi směrech začínaje ve středu obrazu hledáním neizolovaného černého bodu a postupným posunováním se ve směru hodinových ručiček po černých bodech. Hledání je dokončeno ve chvíli, kdy je dosaženo výchozího bodu.



Obrázek 3.4: Vizualizace detekovaných defektů na povrchu oceli za pomocí konstrukce map oblastí významných vlastností.[8]

V posledním kroku je provedena analýza kontur, kde si výzkumnici stanovili parametry praskliny jako oblast, jejíž šířka n je v rozmezí 3 až 6 pixelů a její délka je více jak $n + 3$.

Přístup, který v dnešní době nabírá na populárnosti, je detekce defektů na základě detektorů oblastí významných vlastností (OVV). Guan a kol. [21] navrhli postup, jak detekovat defekty na povrchu oceli pomocí map OVV zkonstruovaných pomocí dekompozice Gaussových pyramid. Celou detekci lze provést ve třech krocích – vytvoření proužků obrázku oceli o různých rozlišeních Gaussovými pyramidami, vytvoření mapy OVV na základě středových rozdílů okolí různých úrovní pyramidy a jejich spojením a v posledním kroku vypočtení středních hodnot z maximální hodnoty každého sloupce a řádku, z nichž ta nižší (v řádku či sloupci) je zvolena pro následné prahování a segmentaci. Výsledek operace lze vidět na Obrázku 3.4

Mapy OVV využili také Li a kol. [36], kteří navrhli model detekce defektů textilních tkanin založený na nízkém zastoupení OVV a Zhao a kol. [62], jež vylepšili metody založené na OVV díky zkoumání tekutin používaných při produkci a jejich interferenci s povrchem objektů, které mají být podrobeny detekci.

[8] Převzato z [21].

Kapitola 4

Vizuální kontrola založená na obrazu

Důležitým krokem v oblasti vizuální kontroly je digitalizace obrazu. Tedy jeho taková reprezentace, která dovoluje scénu zpracovat počítačem.

Dále je potřeba s obrazem nějak manipulovat. Pro jeho manipulaci existují knihovny a frameworky, které tuto práci usnadňují a zrychlují.

Následující kapitola představuje obraz jako spojitou funkci a kroky a podmínky pro jeho digitalizaci. Dále tato kapitola popisuje knihovny a frameworky využité pro zpracování obrazu a práci s neuronovými sítěmi.

4.1 Reprezentace obrazu

Obraz lze reprezentovat jako funkci $C(x, y, t, \lambda)$, kde (x, y) jsou souřadnice jasu zdroje obrazu v čase t a vlnové délky λ. Každý fyzický obraz obsahuje nějakou hodnotu pozadí (tedy funkce obrazu je vždy nenulová) a je omezen na svou maximální hodnotu jasu. Předpokládá se tedy [46]:

$$0 < C(x, y, t, \lambda) \leq A \tag{4.1}$$

kde A je maximální intenzita jasu. Pro všechny obrazy se dále předpokládá, že jsou nenulové skrze obdélníkový prostor pro který

$$-L_x \leq x \leq L_x \tag{4.2}$$

$$-L_y \leq y \leq L_y \tag{4.3}$$

Fyzický obraz je dále pozorovatelný jen po konečný časový interval. Tedy nechť

$$-T \leq t \leq T \tag{4.4}$$

V tomto důsledku je $C(x, y, t, \lambda)$ ohraničená čtyř-dimenzionální funkce s vázanými nezávislými proměnnými. Funkce je dále spojitá na svém definičním oboru [46].

Obraz lze dále definovat i statisticky. Odezva intenzity lidského pozorovatele na obrazovou funkci je měřena jako okamžité záření světelného pole, které lze definovat jako

$$Y(x, y, t) = \int_0^\infty C(x, y, t, \lambda)V(\lambda)\, d\lambda \tag{4.5}$$

kde $V(\lambda)$ reprezentuje *funkci spektrální svítivosti* jakožto spektrální odezvu lidského vnímání. Podobně lze reprezentovat odezvu na barevný obraz pro jeho trichromatické složky – červená (R), zelená (G), modrá (B)

$$W(x, y, t) = \int_0^\infty C(x, y, t, \lambda) W_S(\lambda)\, d\lambda \tag{4.6}$$

kde $W \in \{R, G, B\}$ reprezentující spektrální trichromatické hodnoty pro červenou, zelenou a modrou [46].

Dále je zvolena jedna obrazová funkce $F(x, y, t)$ reprezentující obrazové pole ve fyzickém systému. V případě monochromatického obrazu bude zvolená funkce reprezentovat hodnotu jasu. V případě barevného obrazu jednu z trichromatických složek. Funkce $F(x, y, t)$ může i obecně reprezentovat jakýkoli tří-dimenzionální signál [46].

Průměr obrazu za časový úsek v bodě (x, y) je definován jako

$$\langle F(x, y, t)\rangle_T = \lim_{T\to\infty} \left[\frac{1}{2T} \int_{-T}^{T} F(x, y, t) L(t)\, dt \right] \tag{4.7}$$

kde $L(t)$ je váhovací funkce proměnná v čase. Podobně lze definovat průměrnou hodnotu jasu v daném čase danou prostorovým průměrem [46]

$$\langle F(x, y, t)\rangle_S = \lim_{L_x\to\infty L_y\to\infty} \left[\frac{1}{4L_x L_y} \int_{-L_x}^{L_x} \int_{-L_y}^{L_y} F(x, y, t) L(t)\, dx\, dy \right] \tag{4.8}$$

Mnohé systémy nepracují s časovou variabilitou obrazu a časová konstanta tak z obrazové funkce může být odstraněna. V následující kapitolách tak je učiněno.

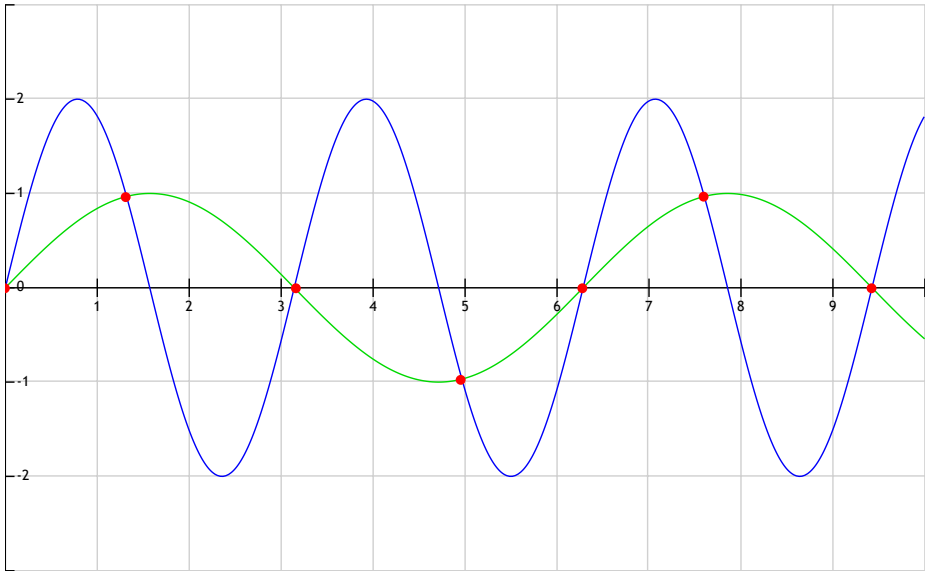
4.2 Definice digitálního obrazu

Digitální obraz $a[m, n]$ popsaný ve dvou-dimenzionálním diskrétním prostoru, vzniká z analogového obrazu $a(x, y)$ definovaného ve spojitém dvou-dimenzionálním prostoru. Odtud vyvstává celý problém. Spojitou funkci analogového obrazu je třeba převést do diskrétního prostoru digitálního obrazu. Tento proces se obvykle nazývá digitalizace a provádí se vzorkováním a kvantováním analogového obrazu [61].

Obraz $a(m, n)$ ve dvou-dimenzionálním spojitém prostoru je vzorkován (rozdělen) do N *řádků* a M *sloupců*. Místa, kde se řádek protíná se sloupcem se nazývají *pixel*, což je nejmenší jednotka obrazu. Hodnoty přiřazené souřadnici (x, y), kde $x \in \{0, 1, 2, \ldots, M-1\}$ a $y \in \{0, 1, 2, \ldots, N-1\}$, procesem kvantování, tvoří výsledný obraz $a[m, n]$ [61].

Diskretizace spojitého prostoru obrazu probíhá **vzorkováním**. Vzorkovací frekvence se řídí „Nyquistovým teorémem“. Ten tvrdí, že vzorkovací frekvence by měla být minimálně dvojnásobkem nejvyšší frekvence zastoupené v signálu (obraze), tedy $f_s \geq 2f_{max}$. V případě dodržení tohoto teorému, nedochází ke ztrátě informace. Není-li tato podmínka dodržena, dochází k tzv. *aliasingu* – překrytí spekter signálu, což vede k jeho nesprávné rekonstrukci (viz Obrázek 4.1) a ztrátě informace [20].

Nelineární transformace mapující spojité hodnoty (úrovně) obrazu do diskrétní množiny se nazývá **kvantování**. Hodnoty funkce jsou rozděleny do úrovní podle rozsahu hodnot. Je-li rozsah 8bitový, lze kvantovat jeho úrovně do 256 hodnot. V případě 16bitového rozsahu až do 65536 hodnot. V průběhu kvantování dochází ke kvantizační chybě, což je rozdíl mezi zaokrouhlenou hodnotou vzorku a jeho reálnou hodnotou. V důsledku kvantizační chyby



Obrázek 4.1: Aliasing vzniká v případě, kdy spojitý vstupní signál (modrý) je rekonstruován (vzorkován) s frekvencí nižší (zelená), než je dvojnásobek nejvyšší frekvence zastoupené v originálním signále. Výsledné zrekonstruované body originálního signálu (červeně).

vzniká kvantizační šum. K vyjádření vztahu mezi rozsahem hodnot pro kvantování a jeho kvantizačním šumem se využívá vzorce $\mathrm{SNR} = -20\log(1/2^n)[\mathrm{dB}]$, kde n je počet bitů pro kvantování [20].

4.3 Nástroje pro práci s obrazem a neuronovými sítěmi

Vývoj aplikací vyšších úrovní umožňují open-source nástroje, které základní stavební kameny již implementují a nabízejí tak možnost stavět komplexnější řešení rychleji a kvalitněji. Následující část nemá sloužit jako vyčerpávající přehled existujících nástrojů. Jsou zmíněny jen nástroje, které byly v této práci využity.

Knihovna OpenCV

OpenCV (Open Source Computer Vision Library) je volně přístupná knihovna pro účely počítačového vidění a zpracování obrazu, jejíž vznik se datuje do roku 1999. Jako taková je napsána v jazyce C a C++ (od verze 4.X.X jen C++) a nabízí rozhraní pro jazyky Java, Python, Matlab a již zmiňované C++. [28]

Knihovna je navržena tak, aby byla dostatečně výpočetně výkonná s ohledem na aplikace běžící v reálném čase. Samotná optimalizace spočívá v upravení algoritmů pro multivláknové zpracování na CPU (optimalizace pro AVX, SSE, MMX, NEON, TBB a OpenMP). Další optimalizace jsou poté umožněny jako Intergated Performance Primitives (IPP) knihovna pocházející od původce samotné OpenCV knihovny – Intel a jako takové existují jen pro procesory tohoto výrobce. V praxi to poté funguje tak, že OpenCV se přímo při běhu rozhodne, jaké instrukce využije k optimálnímu (ve smyslu nejrychlejšímu) vykonání požadované úkonu skrze algoritmus z knihovny. Ani podpora výpočtu na grafických kartách neušla povšimnutí. OpenCV tedy nabízí pro mnoho svých algoritmů možnost přenesení výpočtu na grafickou kartu, umožňujíc tak ještě masivnější paralelizaci. Ty pro karty od

společnosti nVidia pomocí CUDA akcelerovaných výpočtů a pro ostatní grafické karty pak OpenCL optimalizace. [28]

Hlavní výhodou samotné knihovny je její stálý vývoj komunitou programátorů. Do dnešní doby vznikly čtyři verze. První verze je plně s C rozhraním. Tato verze je v této době již považována za zastaralou a není déle podporována komunitou. Druhá verze je jednou z nejpopulárnějších verzí a již využívá výhody jazyka C++. Třetí verze vyniká celkovou obměnou některých modulů. Hlavní výhodou nejnovější čtvrté verze je již plná podpora standardu C++11 a také její plné využití. [28]

Cílem OpenCV je nabízet knihovnu, která je dostatečně jednoduchá na použití a s jejíž pomocí lze vytvářet velké celky. V současné době knihovna disponuje více jak 2500 optimalizovanými algoritmy. Tyto obsahují nejmodernější přístupy v oblasti sledování objektů, klasifikace objektů, extrakce 3D modelů, tvorby 3D shluků bodů, stereo vize, kalibrace kamery, inspekce povrchu a mnohé další. [28]

Každodenně knihovnu využívají společnosti jako Google, Yahoo, Microsoft, IBM, Sony, Honda a další. [28]

Knihovna TensorFlow

TensorFlow je další z volně přístupných knihoven. Tato slouží hlavně pro účely strojového učení. Prvotně vytvořena pro interní využití společností Google, poté, v roce 2015, uvolněna světu. Je napsána hlavně v Pythonu, pro které nabízí rozhraní, dále pak C++ či CUDA. Své rozhraní nabízí též pro jazyk C, JavaScript, Go, C#, Haskell, aj. [1]

Od svého vypuštění došlo k raketovému vývoji. Knihovna tak reflektuje rychlý vývoj v oblasti strojového učení. V současnosti má knihovna za sebou 13 mutací první verze a pracuje se na novém rozhraní verze druhé, která je zatím k náhledu a slibuje více uživatelsky přívětivější rozhraní. [1]

Tvorba modelu v TensorFlow probíhá pomocí grafů. Tvůrce tedy určuje jakým způsobem se data pohybují skrze graf. Každý uzel reprezentuje jednu matematickou operaci a každá hrana spojující dva uzly pak multidimenzionální pole, nazývané též jako *tensor*. [1]

Knihovna je postavena tak, aby pomohla rychle navrhovat celé modely. K tomuto využívá několik úrovní uživatelského rozhraní. Rozhraním na vysoké úrovni je Keras. Pro velké sítě existuje Distribution Strategy rozhraní. Pro okamžité vyhodnocování existuje tzv. Eager rozhraní, které netvoří žádné grafové struktury a vrací přímo hodnoty, čímž napomáhá ladění modelů a celkově sžití se s knihovnou. Mimo tvorbu modelů usnadňuje knihovna i jejich učení pomocí několika metod a portabilitu výsledného modelu. [1]

Alternativami ke knihovně TensorFlow jsou např. PyTorch, Microsoft Cognitive Toolkit (CNTK) či Apache MXNet od Amazonu.

TensorFlow v současnosti používají společnosti jako Airbnb, CoCa-Cola, DeepMind, Twitter a další. [1]

Programovací jazyk C++ a Python

Programovací jazyk **C++** je jedním z objektově orientovaných jazyků. Je považován za jazyk střední úrovně, neboť obsahuje prvky vysokoúrovňových jazyků, přičemž nabízí i nízko úrovňový přístup. Standardní knihovna, která staví na jádře jazyka, nabízí množinu užitečných tříd abstrahujících nízkoúrovňovou logiku, mezi kterou mimo jiné patří správa paměti či způsob její alokace a funkcí, které nabízejí implementace často využívaných algoritmů a ulehčují tak práci. Mezi takové patří např. práce s řetězci (string), poli (array) či vektory (vector). [11]

Počátky jazyka se datují do roku 1979, kdy byl nazýván „C s třídami“. Vznikl jako součást disertační práce Bjarnea Stroustrupa. V roce 1984 pak vznikl první referenční manuál a o rok později také kompilátor Cfront, který vzešel z kompilátoru jazyka C – CPre. Pro jazyk jako takový byl zlomový rok 1990, kdy vznikla ANSI C++ standardní komise a o rok později také ISO C++ standardní komise. V roce 1999 byla členy standardní komise založena knihovna Boost jako inkubátor vysoce kvalitních algoritmů pro standardní knihovny. Rok 1998 dal za vznik standardu C++98 (poslední dvě číslice vždy označují rok vydání), který byl vylepšen roce 2003 standardem C++03 a dlouhou dobu neměl svého nástupce. Začalo se tak mluvit o pomalém úpadku samotného jazyka. Vše změnil standard jazyka C++11, který nabídl obrovské množství nových algoritmů a tříd. Za zmínku stojí představení lambda výrazů, automatická dedukce typu či silně typovaný `nullptr` jakožto náhrada za NULL převzatý z jazyka C. Dále vzniká nový standard každé 3 roky. Nastupoval tedy standard C++14, vedlejší verze standardu, představující mimo jiné generické lambdy. Hlavní verze standardu C++17 nabízí novinky jako je staticky kompilovaná podmínka `if`, pokročilou dedukci datových typů či vložené definice jmenných prostorů. Budoucím standardem bude C++20, jehož největšími změnami mají být koncepty a třícestný operátor `<=>`. [11]

Jazyk C++ je zpětně kompatibilní s jazykem C, v důsledku čehož je mu vytýkána jeho složitost pro začátečníky, která je často způsobena ambiguitou jazyka.

Python je interpretovaný, vysoko úrovňový, dynamicky typovaný jazyk. Jeho autorem je Guido Van Rossum, který vytvořil Python jako nástupce jazyka ABC, ve kterém viděl Rossum problémy a které ve svém návrhu řeší. Python poprvé vyšel v roce 1991 a svůj název dostal dle autorova oblíbeného seriálu „Monthy Python's Flying Circus“. Již ve své první verzi nabízel stěžejní datové typy, např. `list`, `dict` (slovníky) či `str` (řetězce). [23]

Od svého prvotního vydání se těší stálému vývoji. V roce 1994 se dočkal verze 1.0, kde získal funkce jako lambdy, mapy či filtry. Tato hlavní verze se doznala ještě ještě několika vedlejších verzí, kdy končí na verzi 1.6. Verze 2.0, vydána v roce 2000, nabídla možnost vytvoření listu z jakéhokoli iterovatelného typu či automatickou správu paměti (garbage collector). Verze 3.0 představená v roce 2008 upravila stavební chyby jazyka, které nemohly být upraveny bez porušení zpětné kompatibility. V současné době je Python ve verzi 3.8 [23].

Python je v poslední době na vzestupu oblíbenosti mezi jazyky. Nabízí jednoduchost a udržovatelnost výsledného řešení. Stejný kód napsaný v jazyce C++ lze mnohokrát v jazyce Python zvládnout s obrovskou úsporou kódu a vyšší čitelností [23].

V dnešní době, kdy se čím dál více těší oblibě oblast strojového učení, je Python přirozenou volbou pro mnoho vývojářů i začátečníků.

Kapitola 5

Zhodnocení současného stavu a návrh systému pro kontrolu kvality

Následující kapitola obsahuje pohled na současná řešení v oblasti detekce defektů a přesného měření objektů. V návaznosti na to je vytvořena specifikace systému, který je záměrem implementovat. V závěru kapitoly je návrh na řešení takového systému.

5.1 Zhodnocení současného stavu

Oblast přesného (subpixelového) měření nenabízí mnoho možností k velkému zlepšení. Dokazuje to nepříliš vysoký objem odborných prací zabývající se touto oblastí. Prakticky se lze setkat s přístupem, který provádí měření v několika krocích:

1. *Extrakce hrany* – lze volit z několika přístupů. Sobel, Canny, Prewitt, morfologické operace. Je třeba volit na základě dané aplikace, jaký přístup zvolit.

2. *Nalezení hrany se subpixelovou přesností* – tato oblast nabízí několik možností. Jak způsob, jakým body prokládat – křivka, polynom, tak i velikost okolí bodů. Výběr okolí je zase dále závislý na specifikách dané aplikace. Tato část je nejdůležitějším krokem celého měření. Nepřesná detekce hrany vede k dalším nepřesnostem dále v procesu.

   Je možné vybrat si ze tří metod hledání hrany. Interpolační metody nejsou příliš robustní v situacích, kdy je obraz a tedy i hrana zašuměná. Metody založené na momentech zase neobsahují přesné kritérium pro klasifikaci co je hrana a co není. Navíc oblast, kde se integruje musí být v dosahu hledané hrany, jinak tyto metody zcela selhávají. V neposlední řadě metody založené na nejmenší čtvercové chybě jsou odolnější a spolehlivější v případě výskytu šumu, ovšem jejich výpočet může být náročný na výkon.

3. *Proložení hrany* – dále, zase v závislosti na specifikách dané aplikace, dochází k proložení hrany. Tou je velice často kružnice či polynom druhého a vyššího stupně nebo prostá přímka.

4. *Měření* – samotné měření pak probíhá výpočtem vzdáleností jednotlivých proložených hran. V případě kruhu je rozměr znám ihned. Výsledná velikost je dále převedena na jednotky míry dle kalibrace.

Doposud vyvinuté metody pro detekci defektů nevyužívají nejnovější nástroje v oblasti neuronových sítí. Objektivně je to způsobeno rychlostí vývoje oblasti strojového učení. V oblasti měření neexistuje žádná metoda, která by se zabývala přesným měřením střeliva do vzduchových zbraní (dále jen diabolo). Publikované aplikace, které existují (není jich mnoho) v oblasti měření pomocí počítačového vidění buď nevyžadují přesné měření (např. měření papírových krabic) nebo neexistují. Měření rozměrů diabolek a jejich detekci defektů staví nové výzvy.

V závislosti na výše zmíněném jsem se rozhodl vytvořit postupy (algoritmy) pro subpixelové měření rozměrů diabolek. Přesněji její délku (při pohledu zboku) a průměr hlavičky. Dále bych chtěl využít nejnovější poznatky z oblasti konvolučních neuronových sítí k detekci defektů na povrchu samotné diabolky. Ať už podélných rýh na těle diabolky, tak rýh v sukýnce diabolky.

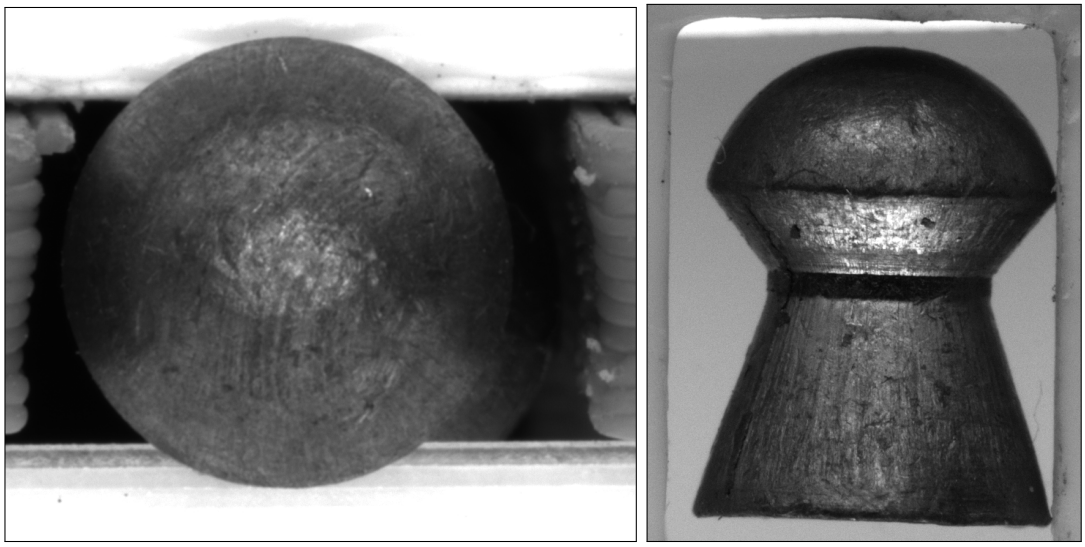
5.2 Specifikace díla

Cílem této práce je vytvořit postupy aplikovatelné v automatické kontrole diabolek. Kontrola je rozdělena na dvě části – část přesného měření rozměrů diabolky ve dvouc pohledech (hlavička a bok) a část detekce defektů diabolky pomocí konvolučních neuronových sítí. V závislosti na specifikách problému jsem se rozhodl pro tuto specifikaci:

- *Použitý vývojový jazyk* – Python, C++.
- *Použité kamery* – 2× Basler acA2440-20gm (hlavička, sukýnka) a 1× Basler acA2040-35gc (pohled zboku)
- *Použité objektivy* – 1× telecentrický objektiv Vico WH03-110CT a 2× bi-telecentrický objektiv Vico DTCM110-80-AL
- *Vstupy* – lokalizované pozice diabolky v obraze (lokalizace není součástí této práce) a kalibrační data.
- *Přístup k detekci defektů* – pomocí konvoluční neuronové sítě.
- *Použité frameworky a knihovny* – Tensorflow [1] (pro neuronové sítě), OpenCV [6] (pro operace z oblasti počítačového vidění).
- *Měřené rozměry* – délka diabolky (pohled zboku) a průměr hlavičky.
- *Přesnost měření* – měření by mělo dosáhnout přesnosti s odchylkou do 0,05 mm pro pohled zboku a s odchylkou do 0,025 mm při měření průměru diabolky.
- *Detekované defekty* – podélné rýhy na těle a vrypy v sukýnce (smajlíci).
- *Přesnost detekce defektů* – přesnost detekce v závislosti na výsledcích v oboru nejlépe 90%.

5.3 Plánovaný postup práce

Oblast detekce defektů objektů prochází s nástupem neuronových sítí postupnou změnou. Z obvykle čistě analytického přístupu detekce defektů, který je drahý na vývoj a není natolik



Obrázek 5.1: Pohledy na diabolku v simulovaném prostředí pro hrubý návrh algoritmů. Vlevo pohled na hlavičku, vpravo pohled zboku.

robustní (často vyžaduje velmi stabilní podmínky), skrze kombinaci analyticky extrahovaných příznaků použitých pro učení neuronové sítě, až po dnešní, čím dál častější aplikace, plně využívající jen neuronové sítě. Tento vývoj umožnil velký pokrok v oblasti neuronových sítí v posledních letech.

Největší otázkou je, jaký přístup z mnoha, jak druhů neuronových sítí, tak dostupných modelů, použít. V oboru lze vidět metody, které využívají přeneseného učení (viz [31]), které je velice užitečné ve chvíli, kdy není k dispozici příliš velká datová sada. Menší nevýhodou takového přístupu je nutnost ctít architekturu výchozí sítě co do vstupních rozměrů.

Další možností je vlastní návrh sítě, kdy analýzou problému a způsobu chování daného objektu, je možné navrhnout síť tak, aby lépe extrahovala takové příznaky, které by se daly od ní čekat. Takový přístup zaujali Chen a kol. [8].

Poslední z hlavních přístupů je využití jednoho z obecných modelů neuronových sítí (AlexNet, GoogLeNet, apod.) a natrénovat celou od začátku. Takový přístup vyžaduje větší množství dat, které se však dá částečně rozšířit tak, že ke každému vstupnímu obrázku jsou vytvořeny ještě další, které budou např. orotované, mít změněn jas či přehozené barevné kanály (probíhá-li učení na barevném datasetu). Výhodou tohoto přístupu je vyšší kontrola nad učením a nízké požadavky na prostředky.

Oblast subpixelového měření má celkem jasnou kostru, kterou je třeba následovat pro úspěšné dosáhnutí cíle. Nejdůležitějším krokem, jak již bylo mnohokrát zmíněno, je subpixelová detekce hrany. K té je možno přistupovat několika způsoby a je třeba vyzkoušet, která metoda bude vyhovovat v daném řešení a podmínkách. V první fázi se zkusí jedna metoda, budou-li výsledky nevyhovující a bude-li vyhodnoceno, že lze získat lepší výsledky s použitím jiné metody, bude využito možností ostatních metod.

Následuje představení navrženého systému z hlediska implementace dle kategorií tak, jak byl problém logicky rozdělen – tedy na kontrolu defektů pomocí neuronových sítí a měření rozměrů diabolky pomocí klasických metod počítačového vidění. Hrubou představu, jak vypadají data v simulovaném prostředí (pro měření) lze získat z obr. 5.1 Na základě těchto obrázků je navržen následující postup.

Návrh koncepce měřících algoritmů

Měřící algoritmy pro oba pohledy – hlavička a zboku budou pravděpodobně vyžadovat různé přístupy.

Z hlediska **pohledu na hlavičku** navrhuji postupovat v těchto krocích:

V prvním kroku bude obraz rozmazán box filtrem, aby došlo k částečnému odstranění šumu. Poté převeden na gradientní. Tento krok pomůže v hledání velkých změn v hodnotách intenzity, což by nasvědčovalo přítomnost hrany.

Vzhledem k temnějším oblastem na stranách hlavičky bude nejprve provedeno hrubé nalezení hrany. V horní oblasti, do určité vzdálenosti od středu obrazu, budou nalezeny hodnoty gradientu. Za účelem omezení množství nalezení zbývajících špiček gradientů, nebude jako hranový pixel brán v potaz ten s největší hodnotou gradientu, ale ten, jehož hodnota je první vyšší než 3/4 nejvyššího gradientu. V těchto oblastech je podezření na přítomnost hrany. Tento „průzkum“ lze provádět s určitým krokem. Stejná sada gradientních špiček bude nalezena na spodní hraně hlavičky.

Těmto nalezeným částem hrany bude dále napasována kružnice a to iterativně tak, aby se odfiltrovaly body gradientu, které jsou příliš mimo průměr. Iterativní pasování kružnice proběhne metodou nejmenších čtverců. Po každé iteraci se zjistí vzdálenost bodů od kružnice. Ty body hrany, které jsou dále než předem zadaná vzdálenost, se nepoužijí pro další iteraci. Ovšem analýza vzdálenosti je po další iteraci provedena i pro ně. Je tedy možné některé body v první iteraci odstranit, aby se nepoužily pro iteraci druhou, ale při splnění kritéria vzdálenosti je použít pro iterací třetí. Tyto iterace pasování kružnice se provádějí tak dlouho, neklesne-li průměrná vzdálenost pod stanovenou hodnotu, není-li změna této průměrné vzdálenosti menší než zadané epsilon či není-li překročen zadaný maximální počet iterací.

Po tomto kroku je tedy známa kružnice, která by měla zhruba opisovat obvod hlavičky za předpokladu relativně přesného nalezení hraničních bodů v prvním kroku a jejich proložení. V dalším kroku proběhně nalezení bodů po celém okruhu. Pro tento účel se obraz převede do polárního souřadného systému podle středu napasované kružnice a rádiusu, který bude zvětšen o konstantu tak, aby se zaručilo zahrnutí hrany jako takové v případě malých nepřesností. Převedení obrazu do polárního souřadného systému pomůže k procházení hrany po obvodu, neboť body obvodové hrany teď budou ležet více méně v jedné vertikální čáře (sloupci obrazu). Hranu tak lze hledat po řádcích ve vzdálenosti kolem rádiusu prvotně pasované kružnice.

V dalším kroku se, teď již po řádcích, zanalyzuje daný obraz. V každém řádku v okolí rádiusu kružnice se nalezne nejvyšší hodnota gradientu. Ten musí mít stoupající charakter na jedné straně a klesající charakter na straně druhé, aby se předešlo detekci šumových hodnot. Stejně tak musí jeho hodnota být poměrově vyšší, než hodnota okolního šumu.

Takto nalezenou hranu lze označit za hranu s přesností atomické jednotky obrazu – pixel. V dalším kroku bude provedeno její subpixelové dopasování. Každý z bodů hrany se proloží polynomem druhého řádu. Pro proložení se použije pět bodů. Maximum proloženého polynomu bude novým bodem hrany se subpixelovou přesností.

Po vypočtení všech proložených hodnot se tyto zpátky přepočtou do souřadného systému původního obrazu. Takto vyextrahované body se znova iterativně proloží kružnicí. Výsledkem takového proložení je rádius a střed kružnice. Rádius bude definovat poloměr dané diabolky s přesností na subpixely.

Při pohledu na **bok diabolky** navrhuji postupovat následovně. V prvním kroku použít medián filtr pro částečné odstranění šumu. Následně obraz převést na gradientní.

Následovat bude detekce bodů hrany v části hlavičky. Postup bude podobný jak u pohledu na hlavičku. Hledání lze omezit na vrchní třetinu obrazu. Hledání gradientu probíhá zleva doprava tak, že se, obdobně, jak u hlavičky, vybere první gradient vyšší, než 3/4 nejvyššího gradientu v hledaném rozsahu.

Vzhledem k požadované přesnosti na měření v tomto pohledu, není třeba provádět detekci hrany se subpixelovou přesností. Postačí čistě její iterativní proložení. Zamezí se tak prokládání bodů, které nebyly zcela přesně nalezeny. Výsledkem bude kružnice se středem a rádiusem.

Pro změření diabolky na její délku není třeba hledat její celkovou konturu. Další hledání se tedy provádí jen v oblasti sukýnky. Postup je prakticky totožný. Obraz se opět prochází zleva doprava a hrana se v tomto případě hledá odspodu nahoru s rozsahem 1/5 celkové výšky obrazu. Dále by se spodní hrana diabolky neměla nacházet.

S přihlédnutím k naklonění diabolky (osa diabolky není vždy rovnoběžné ke kameře), nelze prokládat spodní hranou čistě přímku, ale spíše křivku. Tato křivka může být přesněji polynom druhého stupně – parabola. Pasování samotné paraboly taktéž probíhá iterativně za účelem odfiltrování bodů, které nenáleží hraně. Bude se jednat hlavně o okrajové body obrazu.

Zbývá vypočítat za pomocí obou nalezených primitiv výslednou délku diabolky. V prvním kroku je třeba nalézt naklonění samotné diabolky. V horní části diabolky nám postačí střed napasované kružnice. Ve spodní části je třeba střed získat. Toho lze dosáhnout nalezením krajních bodů spodní hrany. Střed takto definované přímky lze použít jako druhý bod. Vektor definovaný těmito dvěma body vyjadřuje směr náklonu diabolky. Průnik tohoto vektoru a od středu obrazu vzdálenějšího průniku s kružnicí definuje vrchol diabolky. Průnik téhož vektoru a paraboly definuje vrchol na sukýnce. Vzdálenost těchto bodů určuje výslednou délku diabolky.

Návrh postupu při detekci defektů diabolky

V případě návrhu postupu detekce defektů pomocí neuronové sítě nelze navrhnout přímo řešení, bez předešlého vyzkoušení. Navíc ještě ve chvíli návrhu není možné získat rozsáhlé množství dat, které by bylo použito pro učení a testování navržených řešení.

Na základě existujících řešení hledání defektů na povrchu objektů za pomocí konvolučních sítí lze navrhnout, vyzkoušet, některá z řešení. Jedním z nich je možnost přeneseného učení. Toto učení nepotřebuje širokou datovou sadu a je efektivní. Druhým přístupem by mohlo být učení neuronové sítě zcela od počátku. V nejnovějších žebříčcích se z hlediska přesnosti na obecných datasetech pohybuje vysoko neuronová síť GoogLeNet (Inception), která by při dostatečném množství dat mohla přinést i přesnost vyšší než 90 %.

Data pro neuronovou síť je v plánu upravit tak, aby samotná neuronová síť byla robustní. Učit je tedy v plánu i na obrázcích diabolek, které jsou orotované o nějaký stupeň od vertikální osy. Taktéž na snímcích se změněným globálním jasem jak sníženým, tak zvýšeným. Další možností je učení pro různé pozice – tzn. posouvání diabolky v okně.

V případě použití již předtrénované sítě, bude potřeba obrázek upravit i tak, aby odpovídal požadovaným vstupním velikostem. Tento přístup je nejjednodušší co se týče na přípravu učící sady. V mnoha případech je i plně dostačující, když se vstupní obrázek čistě zmenší/zvětší tak, aby požadavky na vstupní velikost splňoval. V případě, že dojde k závěru, že takové učení není dostatečné, bude se muset přistoupit k řešení, které zahrnuje dekompozici vstupu na menší a jejich ruční rozdělení na správné a vadné kusy.

Cílem je vyzkoušet pro daný klasifikační problém jak síť GoogLeNet (Inception v1), tak sít VGG-16, a to s třemi přístupy k učení – zcela od začátku a přenesené učení s doladěním a bez něj.

Kapitola 6

Implementace navrhovaného řešení

I průběhu implementace se postupovalo v dříve navrženém postupu. Odchylky od návrhu implementace byly nutné ve chvíli, kdy to data z reálného prostředí vyžadovala. Následující kapitola je rozdělena na analytickou část, popisující postup implementace měřících modulů a část detekující defekty pomocí metod strojového učení.

Implementované postupy mají provést celkovou kontrolu diabolky. Ta probíhá ve dvou, na sobě nezávislých částech. Tedy část detekující defekty může probíhat paralelně s měřícími úlohami. Obrázek 6.1 zobrazuje jednotlivé bloky.

6.1 Moduly měřící délku a průměr diabolky

Všechny měřící moduly začínají detekcí hrany s přesností na pixel a končí samotným výpočtem rozměru. Postup práce lze názorně vidět na Obrázku 6.1. Jednotlivé bloky jsou rozebrány dále.

Detekce hrany s pixelovou přesností

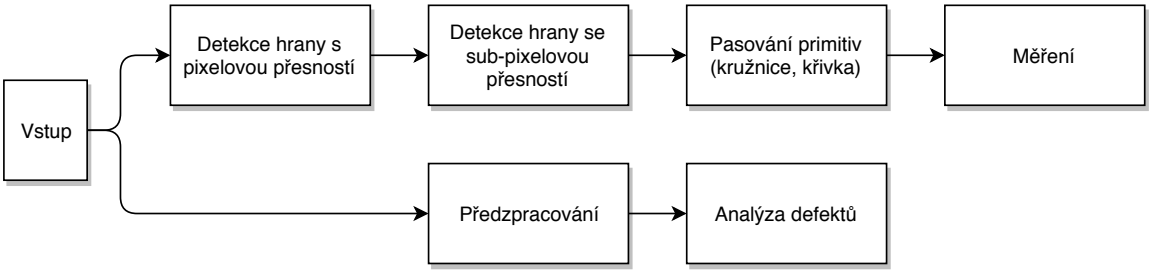
Detekce hrany se provádí na gradientním obrazu vstupu (viz Obrázek 6.2[1]). Pro **pohled zboku** (měření délky) se detekuje hrana jen v horní oblasti (hlavička) a spodní oblasti (sukýnka).

Hledání hrany probíhá s krokem 1 pixel, tedy hledá se po celé šířce obrazu, ovšem co se týče hloubky prohledávání, výšky obrazu, hledá se jen do 2/7. Tento parametr se ukázal jako nejlepší. V této části se očekává, že se horní část diabolky nachází. Není tedy třeba hledat po celé výšce. Snižuje se tak počet špatně nalezených hranových bodů. Pro hledání hranových pixelů se ukázalo nejlepší prohlásit za hranový takový první pixel, jehož hodnota je vyšší, než 3/4 nejvyšší hodnoty v hledaném rozsahu.

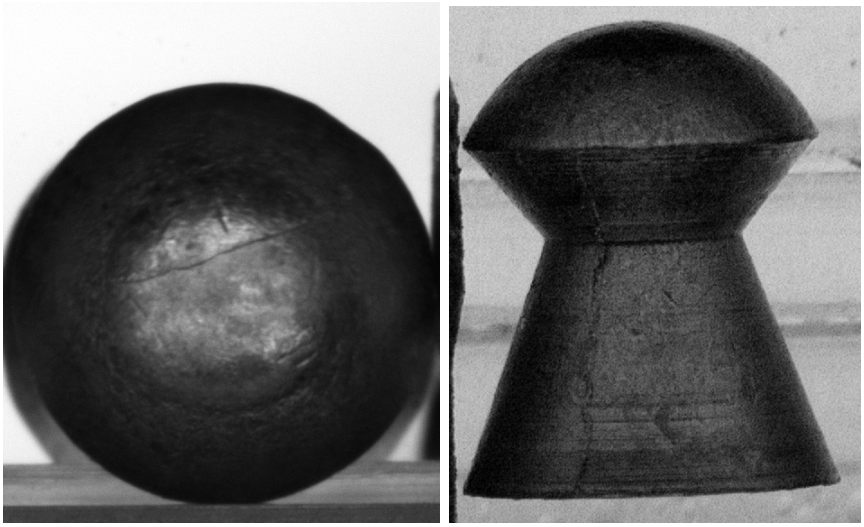
Takto detekovaná hrana obsahuje některé body, které na první pohled nejsou správné. Počet těchto bodů je však minoritní a budou odstraněny v dalších krocích. Stejný postup je použit pro spodní část diabolky (oblast sukýnky). Se stejnými parametry, až na množství prohledávané části, co se týče do hloubky obrazu. Tvar sukýnky, který je mírně prohnutý, umožňuje omezit hledání ještě více. Prohledává se tedy jen 1/5 obrazu co do výšky. Rozsah na šířku zůstává.

Při **pohledu na hlavičku** je postup velice podobný. Vstupem do metody je přibližná pozice středu diabolky. Prvotní hrana je ze spodní strany hledána nikoli v celém rozsahu

[1]Obrázky jsou upraveny pro lepší viditelnost příznaků. Všechny následující vizualiace metod jsou na neupravených obrázcích.



Obrázek 6.1: Jednotlivé bloky popisující postup při celkové kontrole diabolky. V měřících modulech (horní část) je při pohledu zboku vynechána sub-pixelová detekce hrany. Výsledkem měřících modulů je změřená hodnota délky a průměru hlavičky diabolky. Z modulu pro detekci defektů pak případný defekt diabolky. Výstup z těchto modulů je poté použit pro vyhodnocení, zda-li je diabolka vadná či nikoli.

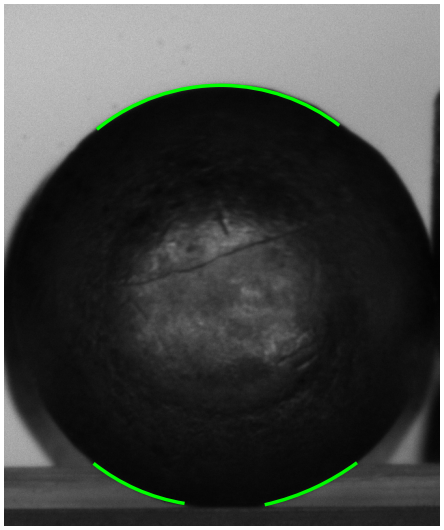


Obrázek 6.2: Příklad vstupních obrázků pro reálný systém. Pohled na hlavičku (vlevo) a boční pohled (vpravo).

obrazu, ale jen ve vytyčených rozsazích od středu. Což je výsledek řešení problému, který na testovacích vstupech při návrhu metod nebyl. Hrana hlavičky v místech dotyku s podkladem není dostatečně definována. Tento rozsah je omezen na ±200 px od středu diabolky s vypuštěním cca 140 px přímo ve středu. Hledá se tedy jen v místech, kde je jistota nalezení solidní hrany, tedy takové, jíž hodnota gradientu vysoce převyšuje hodnoty okolních gradientů. Velice dobrá viditelnost hrany umožňuje prohlásit za hranových pixel takový, jehož hodnota je nejvyšší v hledaném rozsahu (sloupci). Hloubka hledání je omezena na 1/5 výšky obrazu.

Z vrchní strany se hrana hledá taktéž jen ve vytyčeném (±200 px od středu diabolky) rozsahu co do šířky obrazu a také co do výšky obrazu. Koeficient pro prohlášení pixelu za hranový je nastaven na 3/4 nejvyššího v prohledávané oblasti (1/4 výšky obrazu). Oblasti použité pro prvotní hledání hrany jsou zvizualizované v Obrázku 6.3.

Takto nalezené hrany jsou poté proloženy kružnicí iterativně, aby došlo k odfiltrování bodů hrany, které se nesprávně detekovaly. Parametry prokládání, které po několika pokusech vykazují nejlepší výsledky, jsou nastaveny následovně:



Obrázek 6.3: Oblasti použité pro prvotní hledání hrany (zelená). Hrana nalezená v těchto místech je poté prokládána kružnicí, která slouží jako prvotní odhad pozice hrany.

- Maximální počet iterací: 10.
- Epsilon: 1e−3.
- Maximální vzdálenost bodu od kružnice: 3.
- Minimální počet bodů použitý v jedné iteraci: 50%.

Pokud proložení není úspěšné do deseti iterací, nedá se očekávat jeho úspěšnost v iteracích dalších, proto tato pojistka.Parametr maximální vzdálenosti je využit pro prokládání mezi iteracemi. Po prvotním proložení se z množiny bodů použité pro další proložení použijí jen ty, které splňují danou podmínku. V případě, že podmínku v některé z dalších iterací splní, jsou tyto vyřazené body znovu použity pro další pasování.

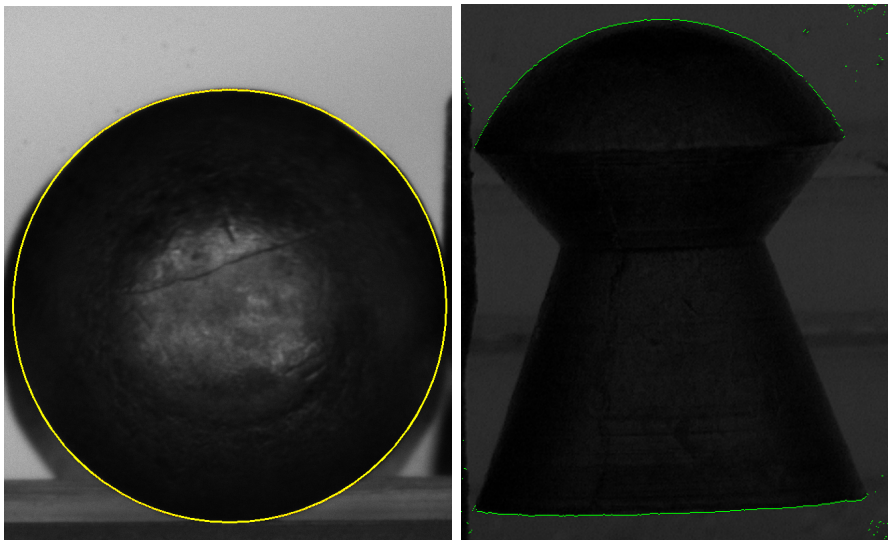
Důležitým parametrem je také minimální počet bodů použitý pro pasování. V případě, že postupným odebíráním bodů se algoritmus dostane do situace, kdy by použil méně, než polovinu vstupních bodů, je toto pasování prohlášeno za neúspěšné. K těmto případům dochází v situacích, kdy detekovaná hrana (její body) nedostatečně reprezentují očekávaný tvar. Z hlediska algoritmu to není chyba, neboť tak nepřímo detekuje zmetkové diabolky, které mohou být ihned vyřazeny.

Výsledek detekce hrany s pixelovou přesností pro oba pohledy lze pozorovat v obrázku 6.4.

Detekce hrany se sub-pixelovou přesností

Hledání hrany se sub-pixelovou přesností se provádí jen na pohledu na hlavičku. Při pohledu zboku je dostačující její parametrizace (proložení) prováděné v dalším kroku.

Jako vstup pro detekci sub-pixelové hrany při pohledu na hlavičku se opět používá gradientní obraz. Tento obraz je převeden do polárního souřadného systému s lineárními vzdálenostmi vzdálenostmi (dále polární souřadný systém, viz Obrázek 6.5). Pro tento převod je právě využita kružnice z předešlého kroku, která hrubě definuje oblast, kde se s největší pravděpodobností nachází hrana hlavičky diabolky.



Obrázek 6.4: Detekce hrany s pixelovou přesností. Pro hlavičku (vlevo) definuje počáteční hranu kružnice (prokládané body, červeně, proložená kružnice, žlutě). Pohled zboku (vpravo) ukazuje počáteční množinu bodů ve zkoumaných oblastech – sukýnky a hlavičky.

Díky reprezentaci hlavičky diabolky v polárním souřadném systému lze hranu hledat spíše jak na kružnici, ve sloupci. Samotné vyhledávání tedy spočívá v procházení obrazu po řádcích a hledání vrcholků gradientů v rozmezí kolem rádiusu kružnice, na základě které byl převod proveden.

Po nalezené maximální hodnoty gradientu v prohledávaném rozsahu, jsou jeho okolní body proloženy polynomem druhého řádu. Jednoduchým výpočtem (viz rovnice 2.11) je pak nalezen vrchol paraboly (potřebná je jen x-ová souřadnice), který určuje sub-pixelovou pozici gradientu. Výsledná množina takto nalezených bodů je převedena do kartézského souřadného systému (viz Obrázek 6.5), kde je připravena k dalšímu kroku.

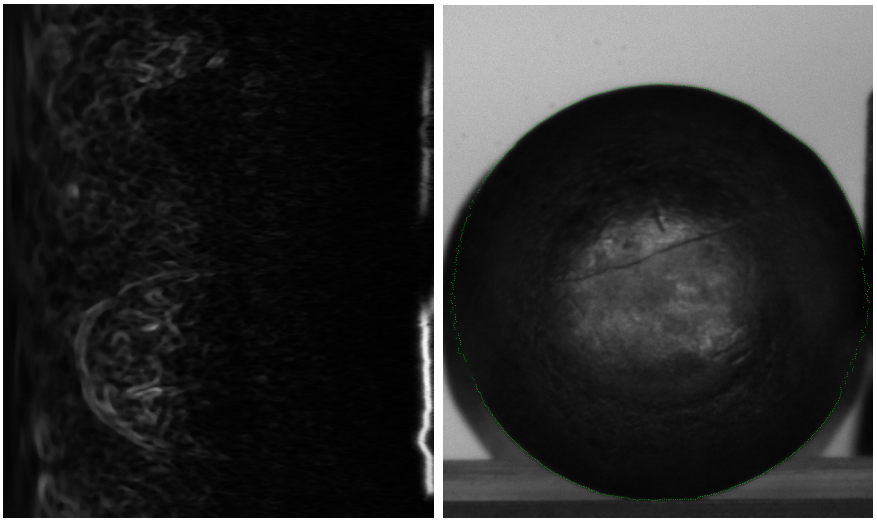
Proložení primitivy

Při pohledu na hlavičku se finální hranou prokládá jen kružnice a to metodou nejmenších čtverců. Probíhá tak opět iterativně za stejných podmínek až na jednu, a tou je maximální vzdálenost bodu od pasovaného primitiva. V tomto případě kružnice. Jelikož kružnice už samotná se pasuje na sub-pixely, lze tuto vzdálenost definovat jako desetinné číslo. Pro toto pasování se ukázalo jako nejlepší parametr $1,5$ pixelu. Tedy aby daný bod byl použit pro další iteraci pasování, nesmí jeho vzdálenost od okraje kružnice překročit tento parametr. Tato restrikce je použita bez omezení na směr, ve kterém k odchýlení dojde (od/k středu).

V případě pohledu zboku se v oblasti hlavičky prokládá kružnice úplně stejně jako pro samotný pohled na hlavičku. parametr maximální vzdálenosti při prokládání je nastaven na 1.

Ve spodní části (sukýnka) se hrana prokládá polynomem druhého řádku (parabolou). Vzhledem ke specifikám v oblasti sukýnky (vyšší stupeň roztřepenosti), se osvědčil také jiný parametr pro vzdálenost bodu od paraboly mezi iteracemi. Pro tento případ 2 pixely.

Napasovaná primitiva pro oba pohledy lze vidět na Obrázku 6.6.



Obrázek 6.5: Reprezentace hlavičky v polárním souřadném systému (vlevo). Převedení proběhlo s využitím parametrů kružnice získanými v předchozím kroku. Prohledávaná oblast je u pravého okraje. Výsledné body hrany (vpravo). Lze sledovat menší nepřesnosti ve tmavých oblastech na krajích hlavičky.

Měření

Předpokladem pro měřící moduly je zkalibrovaný systém. Kalibrace není součástí této práce, ovšem pro úplnost zde je zmíněn postup, jakým ke kalibraci dochází. Kalibr ve tvaru diabolky je změřen měřícími moduly a to několikrát. Z protokolu, který je k danému kalibru vypracován je odečtena reálná hodnota rozměru diabolky pro daný pohled. Ze změřeného rozměru v pixelech a hodnoty reálného kalibru je vypočtena velikost pixelu.

Průměr hlavičky je dán přímo průměrem napasované kružnice. U pohledu zboku, tedy hledání délky diabolky, je třeba již z proložených primitiv (kružnice a parabola) nalézt jejich vrcholy, které výslednou délku reprezentují (viz Obrázek 6.7).

Od navrženého postupu se implementace odchyluje v použití bodu pro definici naklonění ve spodní části diabolky. V prvním kroku se definuje úsečka (U) vzniklá z krajních bodů paraboly (P) napasované v části sukýnky. Mějme dále přímku (L), která je rovnoběžná s úsečkou U a je tangentou k parabole P. Bod dotyku této tangenty je hledaným počátečním bodem v části sukýnky. Nechť U je definována jako $f(x) = a_1x + b_1$, L jako $f(x) = a_2x + b_2$ a P jako $f(x) = dx^2 + ex + f$. Za předpokladu, že výsledná přímka L má shodný náklon $(a_1 = a_2)$, zbývá vypočítat hodnotu parametru b_2. Všechny ostatní parametry jsou známé. Postup výpočtu je následující:

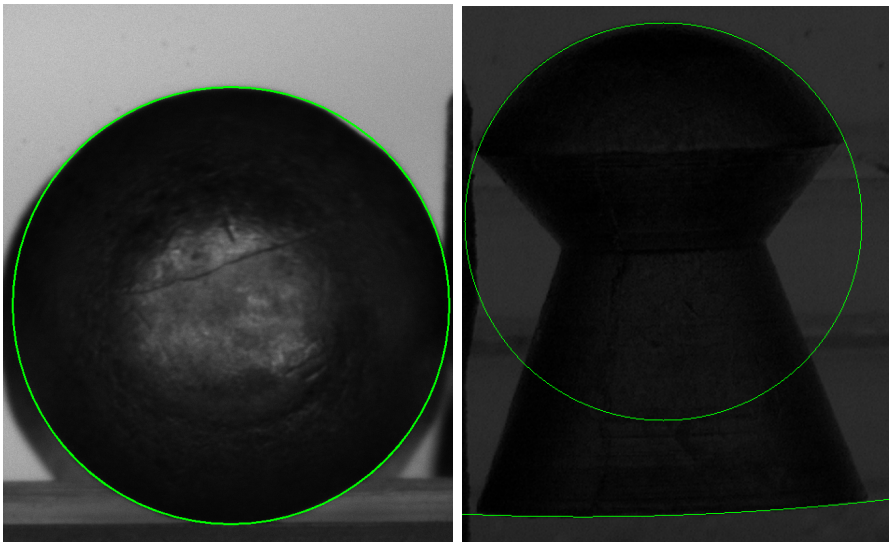
$$a_2x + b_2x = dx^2 + ex + f \tag{6.1}$$

$$0 = dx^2 + (e - a_2)x + (f - b_2) \tag{6.2}$$

$$D = B^2 - 4AC,\ B = e - a_2,\ A\ = d,\ C = f - b_2 \tag{6.3}$$

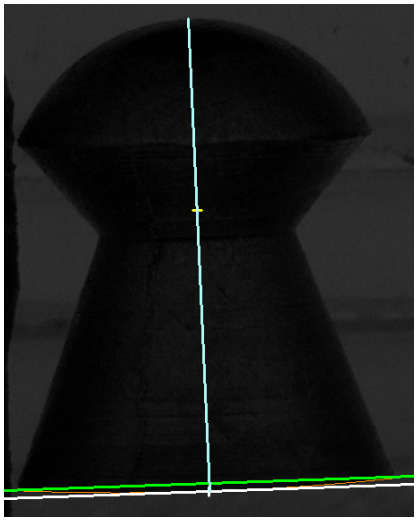
$$D = 0 \Rightarrow \text{jediné řešení} \tag{6.4}$$

$$b_2 = \frac{4df - (e - a_2)^2}{4d} \tag{6.5}$$



Obrázek 6.6: Výsledek pasování primitiv. Pro pohled na hlavičku (vlevo) je to kružnice. Pro pohled zboku (vpravo) je to v horní části kružnice a ve spodní části parabola.

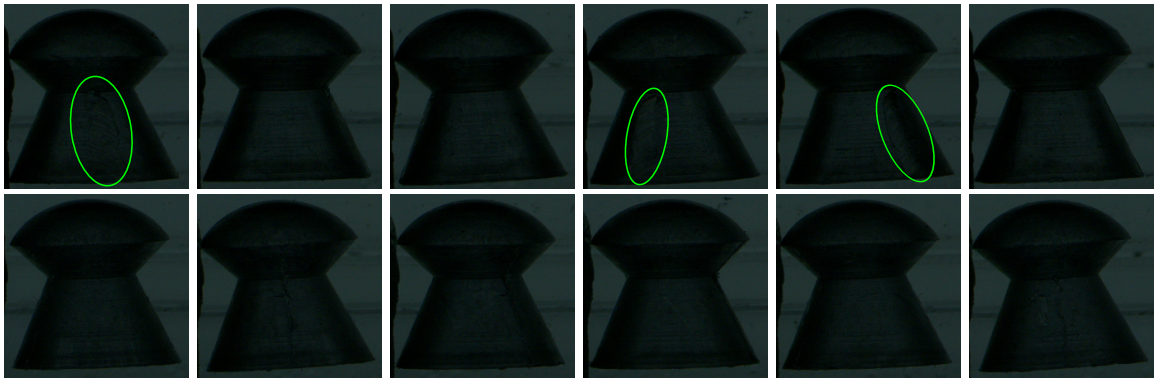
Bod dotyku tangenty s parabolou a střed kružnice definují vektor, který je pak převeden na jednotkový. Násobením jednotkového vektoru rádiusem kružnice lze získat bod na kružnici a po přičtení souřadnic středu kružnice přímo bod v obrazu v oblasti hlavičky. Nalezené body z obou oblastí (hlavičky a sukýnky) definují výslednou délku diabolky (kód viz Zdrojový kód 6.1).



Obrázek 6.7: Postup odečtu výsledné délky diabolky. Úsečka vytvořená z krajních bodů paraboly (zelená), přímka k ní paralelní s bodem (křížem) dotyku tangenty (bílá) s parabolou (tmavě oranžová), střed kružnice (žlutá) a výsledná úsečka definující délku diabolky (světle modrá).

```
Parabola parabola{ edgeExtractorData.getParabolaSkirt() };
vector<double> parEq{ parabola.getParabola() };
double imWidth{ edgeExtractorData.getImageWidth() };

// úsečka definující náklon
double yLeft{ getPolyVal(0., parEq) };
double yRight{ getPolyVal(imWidth, parEq) };

Line equation{ createEqu(Point2d{0., yLeft}, Point2d{imWidth, yRight}) };
Line parTangent{ findParalellTangentToLine(Line{equation}, parabola) };

// výpočet bodu průniku tangenty a paraboly
double xTangentTouch{ (-(parEq[1] - parTangent.getLine()[1]))
    / (2 * parEq[2]) };
double yTangentTouch{ getPolyVal(xTangentTouch, parEq) };

Point2d intersectionBot{ xTangentTouch, yTangentTouch };
Circle circle{ edgeExtractorData.getCircleHead() };
Line2D centralLine{ intersectionBot, circle.getCenter() };
Point2d intersectionTop{ Point2d{circle.getCenter()
    + centralLine.dir * circle.getRadius() };
```

Zdrojový kód 6.1: Ukázka kódu pro výpočet průsečíku s hlavičkou a sukýnkou při pohledu zboku.



Obrázek 6.8: Příklad vstupních obrázků (šestice) pro neuronovou síť klasifikující vadné kusy s vyznačenými vadami (podélné rýhy, horní řada) a správné kusy (dolní řada). Vstupní data jsou ve velikosti 224×224 pixelů.

6.2 Neuronová síť pro detekci defektů diabolky

Výběr a výkon neuronové sítě pro řešení problému ovlivňuje několik faktorů. Prvním krokem je výběr modelu, který bude použit pro daný problém, druhým je příprava datové sady pro daný model a upravení architektury modelu tak, aby reflektoval zvolený způsob učení. V neposlední řadě je to způsob učení a nastavení parametrů tak, aby výsledná síť podávala nejlepší výsledky.

Datová sada

Pro kontrolu defektů pomocí neuronové sítě byly vybrány defekty objevující se na pohledu zboku a ze spodní strany diabolky. Z boční strany jsou to podélné rýhy, ze strany spodní jsou to rýhy v sukýnce, též nazývané jako smajlíci.

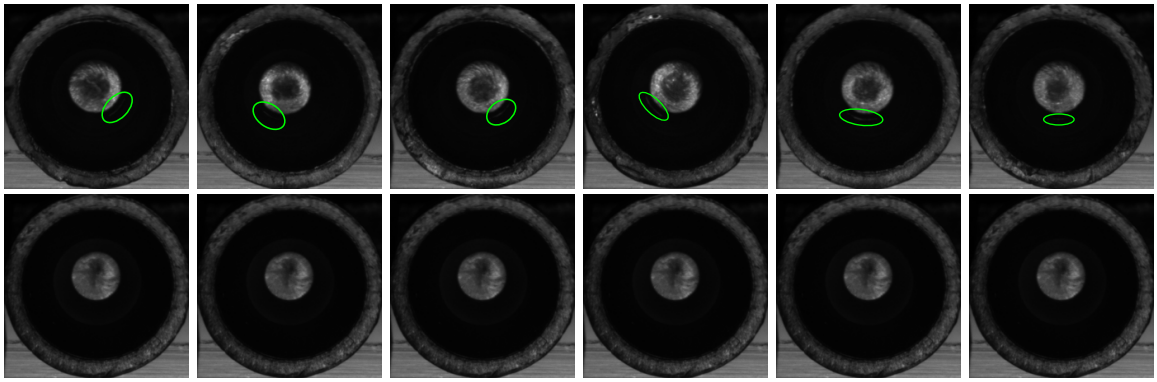
Pro pohled zboku je nutné analyzovat diabolku po celém svém obvodu. Systém sběru dat je tedy navržen tak, že pro učení reprezentuje jednu diabolku vždy šestice. Obrázky jsou jednotlivě pořízeny, zatímco se diabolka odvaluje po dráze. Výsledkem je sada obrázků, která po částech tvoří jeden celek. Neuronová síť se tedy musí vypořádat se vstupem pro šestici obrázků. Architektura sítě tento fakt musí reflektovat. Příklad vstupních obrázků pro pohled zboku lze vidět na Obrázku 6.8. Vstupní obrázky jsou tříkanálové.

Při pohledu zespodu diabolky se kontroluje výskyt „smajlíků“. Vstupem pro neuronovou síť je ořízнutý obrázek, který vznikl předzpracováním. Rozdíl oproti pohledu zboku je, že jeden obrázek definuje jednu diabolku, snímek je dále pouze černobílý. Příklad vstupních dat lze vidět na Obrázku 6.9.

Augmentace dat nebyla provedena. Okolní podmínky co se týče rotace, posuvu či jasu jsou velice stabilní, proto „umělá“ augmentace není potřeba. V případě pohledu zboku navíc do obrazu zasahují částí vozíčku, který zajišťuje pohyb diabolky pod kamerou. Právě rotace či různá převracení by mohla mít spíše neblahý dopad na výsledný výkon neuronové sítě.

Modely využité pro klasifikaci

Pro problém detekce defektů byly zvoleny dva modely neuronových sítí, ze kterých se vybere ta, která bude mít lepší výsledky. Prvním z nich je model VGG-16, druhým je Inception v1. Volba padla právě na tyto dva, neboť VGG modely jsou v oblasti detekce defektů používány



Obrázek 6.9: Příklad z datové sady vstupů pro neuronovou síť klasifikující vadu „smajlíci“ (vyznačeno, horní řada) a bezvadné kusy (dolní řada). Vstupní data jsou ve velikosti 224×224 pixelů.

běžně a Inception je model v současnosti vykazuje jedny z nejlepších výsledků. Obrovský rozdíl je také v počtu parametrů, které daný model má. Inception model ve své verzi 1 ladí kolem 5 milionů parametrů, kdežto VGG-16 něco málo přes 138 milionů. Tyto faktory se odráží ve fyzické velikosti výsledného natrénovaného modelu a také v rychlosti výsledné klasifikace.

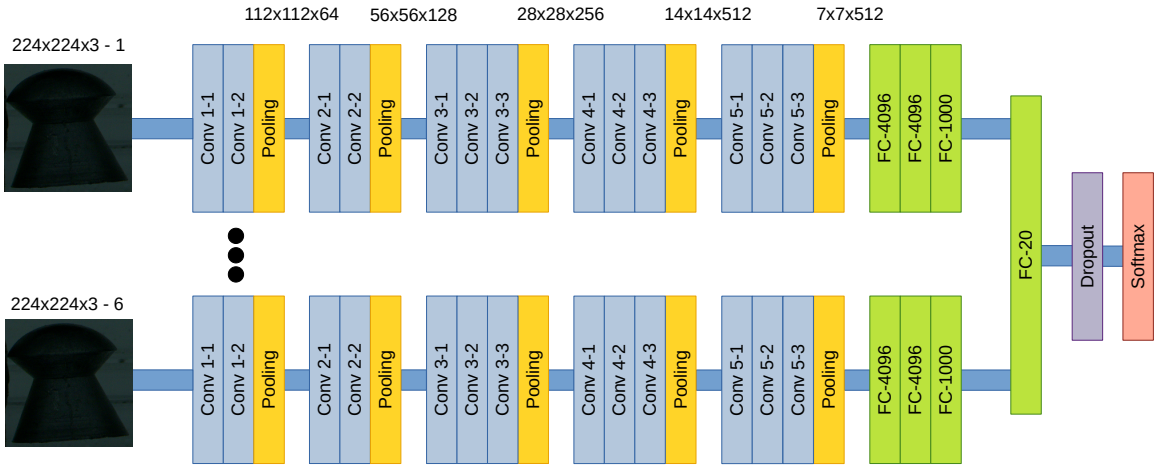
Oba modely byly využity shodně. Pro pohled zboku, který jednu diabolku reprezentuje šesticí tříkanálových obrázků je šest sítí spojeno dohromady (viz Obrázek 6.10). Tedy jedna síť pro jeden obrázek. Výstupy z plně propojených vrstev jsou poté zastřešeny poslední plně propojenou vrstvou. Tato vrstva klasifikuje do dvaceti tříd. Volba takového množství výstupů je ze dvou důvodů. Prvním z nich je, že nabízí jednoduchou rozšiřitelnost do budoucna, bude-li potřeba, aby stejný model klasifikoval ještě další defekty. Druhý důvod je, že neuronová síť, která klasifikuje do malého počtu tříd (v tomto případě by to byly jen dvě) nevykazuje tak dobré výsledky jako síť, která má v plně propojené vrstvě větší množství výstupů, i přesto, že se jejich váhy přímo neučí.

Pro pohled ze spodní strany je použita architektura modelu tak, jak je, bez dalších úprav. Jediná úprava je ve vstupních datech. Jelikož oba modely (VGG i Inception) vyžadují tříkanálový vstup, je šedotónový vstup ze spodního pohledu ztrojen. Tento krok je nutný pro případ, kdy se využívá síť již předučená. Architektura sítě je jen pro úplnost na Obrázku 6.11.

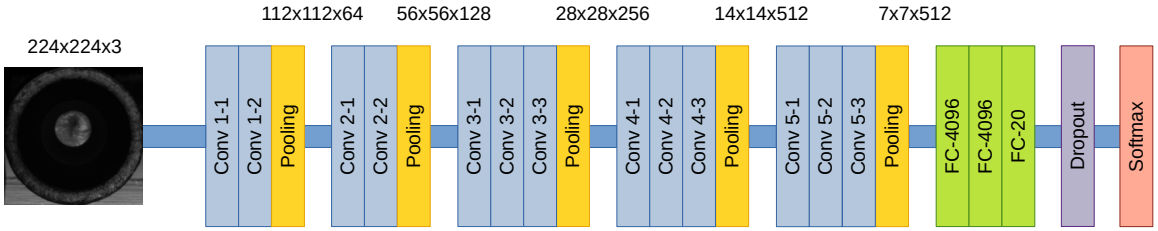
Přístupy k učení modelů

Učení probíhalo na datové sadě, která čítala cca 500 vzorků od každé kategorie. Tedy v případě pohledu zboku to byly podélné rýhy a bezvadné kusy, v případě pohledu zespodu smajlíci a bezvadné kusy. Celá datová sada byla rozdělena na testovací sadu a učící sadu. Postup učení se poté vyhodnocuje dle úspěšnosti klasifikace pro testovací sadu. Dělení je v poměru 1:10. Tedy desetina celkové datové sady slouží jako testovací.

V případě obou sítí byly zvoleny tři přístupy k učení. První přístup je učení celé sítě s náhodně nastavenými váhami. Tedy pokus, je-li se síť schopna naučit na takovém množství dat. Druhý přístup je přenesené učení, kdy se předučené síti nahodně nastaví jen poslední plně propojená vrstva. V případě neuronové sítě pro pohled zboku je to poslední plně propojená vrstva každé jednotlivé sítě a ta úplně poslední, do které se agregují výstupy z jednotlivých sítí. Celá síť jinak pokračuje v učení tak, jak byla předučená na datové sadě



Obrázek 6.10: Architektura sítě pro detekci defektů při pohledu zboku. Základ v tomto případě tvoří model VGG-16, který je zvlášť pro každou jednu část z šestice pohledů. Jejich výsledek se poté spojuje jednou plně propojenou vrstvou (FC). V případě modelu Inception byl zvolen stejný přístup.



Obrázek 6.11: Architektura sítě pro detekci defektů při pohledu zespodu. Základ tvoří opět model VGG-16, jehož změna tkví v poslední plně propojené vrstvě. Opět, v případě modelu Inception byl zvolen stejný přístup.

ImageNet. V posledním případě je vyzkoušeno učení takové, kdy se opět váhy náhodně nastaví v poslední plně propojené vrstvě, ale učení probíhá bez ladění konvolučních vrstev sítě. Učí se tak jen poslední plně propojené vrstvy.

Postupnými pokusy a výsledky z předešlých zkušeností a doporučení autorů modelů, byly sítě učeny s následujícími parametry. Učení probíhá, v případě přeneseného učení, s maximem devíti kol. Každé kolo obsahuje 1000 epoch. Učení na kola je zvoleno z toho důvodu, neboť tento přístup dovoluje měnit učící podíl (learn ratio) mezi koly, a to dle potřeby a vývoje učení. Učící podíl začíná na hodnotě 0.01 pro urychlení učení v počátcích. Po třech kolech se snižuje na hodnotu 0.001 a po dalších třech kolech dál na 0.0001, tak jak se neuronová síť blíží k „řešení“. Učení je ukončeno ve chvíli, kdy se zjistí, že síť se ustálila a již není schopna dalšího zlepšení. V případě učení neuronové sítě od úplného začátku (bez využití přeneseného učení) se učící podíl mění stejně, ale po vícero kolech. Přesněji ve 25. a 28. kole pro síť VGG16 a 10. a 13. kole pro Inception. Tyto čísla nejsou vybrána náhodně. V případě učení celé sítě bez využití přeneseného učení, je to právě 25. kolo (resp. 10.), kdy neuronová síť vykazuje známky zlepšení. Obvykle poté stačí maximální celkový počet kol 35 (resp. 20). Velikost učícího vzorku (batch size) je 8. Posledním důležitým parametrem je koeficient zahozených vah (dropout), který je nastaven na 0,2.

Kapitola 7

Vyhodnocení implementovaného řešení

Každý vytvořený systém či řešení by mělo být otestováno tak, aby bylo prokazatelné splnění vstupních požadavků. Systémy lze testovat z mnoha hledisek, ovšem pro implementované metody v této práci dává smysl jejich vyhodnocení z hlediska přesnosti a opakovatelnosti. Parametr opakovatelnosti je obtížné vyhodnotit, neboť vyhodnocení probíhá v reálném prostředí. V tomto je diabolka zachycena v pohybu a její pozice a hlavně naklonění k ose kamery je vždy mírně odlišné. Parametr opakovatelnosti je třeba brát s ohledem na tuto skutečnost.

Pro neuronové sítě je vyhodnocena jejich rychlost vyhodnocení. Dále pak metriky kombinací míry falešné pozitivity (false positive, dále jen FP), míry falešné negativity (false negative, dále jen FN) a míry pravdivě pozitivních (true positive, dále jen TP) – přesnost, sensitivita (recall) a F-skóre.

Všechny výkonnostní testy byly vyhodnocovány na sestavě s CPU Intel Core i5 3,2 GHz s 16 GB paměti a grafickou kartou GTX 1080 Ti.

7.1 Vyhodnocení měřících modulů

Samotné ověřování přesnosti měření probíhá následovně. Diabolka je nejprve změřena na zkalibrovaném nástroji pro přesné měření rozměrů a poté je změřena pomocí měřícího modulu. Rozdíl v těchto dvou naměřených hodnotách vyjadřuje přesnost daného měřícího modulu. Vzhledem k náročnosti tohoto postupu, bylo v tomto kroku vyhodnoceno 15 diabolek.

Opakovatelnost měření je vyhodnocena následovně. Do stroje je založeno 15 diabolek. Měřící moduly těchto 15 diabolek změří 10krát. Každá diabolka tedy desetkrát projede strojem. Pro každou diabolku je poté vypočtena směrodatná odchylka měření (σ) a pro celou sadu patnácti diabolek je poté spočtena kombinovaná směrodatná odchylka. Výpočet kombinované směrodatné odchylky je umožněn z toho důvodu, že jednotlivé podskupiny lze sloučit do jedné skupiny. Tyto podskupiny tvoří homogenní celek [3].

Posledním vyhodnocovaným parametrem je rychlost daného algoritmu. Rychlosti změření jednotlivých diabolek pro oba pohledy byly vyhodnoceny změřením 5000 diabolek. Z množiny naměřených hodnot byly poté odstraněny extrémy, které vznikly inicializací metod, a ze zbývajících hodnot spočten průměr.

Pohled	Přesnost	Kombinovaná směrodatná odchylka	Rychlost
Zboku	25 µm	10 µm	4,25 ms
Na hlavičku	10 µm	5 µm	28,3 ms

Tabulka 7.1: Vyhodnocení měřících modulů z hlediska přesnosti, kombinované směrodatné odchylky (opakovatelnosti) a rychlosti změření.

	Parametr	Inception v1	VGG-16
Přenesené učení	Přesnost (%)	100	100
	Sensitivita (%)	100	100
	F-Skóre	1	1
Přenesené učení s doladěním	Přesnost (%)	100	100
	Sensitivita (%)	100	100
	F-Skóre	1	1
Čisté učení	Přesnost (%)	100	100
	Sensitivita (%)	100	100
	F-Skóre	1	1

Tabulka 7.2: Výsledky klasifikace neuronových sítí při různých přístupech učení. Přenesené učení s učením jen posledních plně propojených vrstev, přenesené učení s doladěním i konvolučních vrstev a čisté učení, při kterém se učí neuronová síť úplně od začátku. Výsledky jsou stejné pro oba pohledy.

V případě měření při pohledu na hlavičku, byl postup zcela stejný. Výsledky pro oba pohledy lze vidět v tabulce 7.1.

7.2 Vyhodnocení detekce defektů pomocí neuronové sítě

Jak již bylo zmíněno, neuronové sítě jsou vyhodnoceny z hlediska přesnosti, sensitivy a F-skóre (viz rovnice 7.1 – 7.3). Dále je pak vyhodnocena také rychlost vyhodnocení.

$$přesnost = \frac{TP}{TP + FP} \tag{7.1}$$

$$sensitivita = \frac{TP}{TP + FN} \tag{7.2}$$

$$F\text{-}skóre = \frac{2 \times přesnost \times sensitivita}{přesnost + sensitivita} \tag{7.3}$$

Tyto ukazatele jsou vyhodnoceny pro všechny tři způsoby učení (až na rychlost). Výsledky lze vidět v tabulce 7.2.

Výsledky přesnosti, sensitivity a F-skóre jsou stejné pro oba pohledy nehledě na volbu způsobu učení. Vzniká tedy podezření na přeučení dané neuronové sítě. Z tohoto důvodu byla pořízena další testovací data v objemu asi 300 kusů od každé kategorie. Výsledek byl i poté stejný. Závěr je takový, že neuronová síť má stoprocentní úspěšnost při zachování generalizace. Je možno se tedy domnívat, že v případě dobře definovaných a stabilních podmínek vykazují neuronové sítě vynikající výsledky. Ke stabilitě napomáhá taktéž datová sada. Neobsahuje mnoho tříd (vad), ale množiny správných a defektních vzorků jsou zcela

	Na hlavičku	Zboku
Inception v1	7 ms	31 ms
VGG-16	7,5 ms	42 ms

Tabulka 7.3: Průměrná rychlost vyhodnocení na grafice GTX 1080 Ti pro oba modely neuronových sítí a oba pohledy. Lze pozorovat, že v případě kompozice při pohledu zboku, se ukazují výhody sítě Inception v1.

disjunktní, přičemž obrovské množství dat v obraze je neměnných. Ať už je to samotný tvar diabolky či pozadí. Síť se tak může plně soustředit na malé nuance mezi třídami. Dalším faktorem může být nedostatek dat a je tedy možné, že ve chvíli, kdy daná síť bude nasazena v reálném prostředí, nebude vykazovat tak dobré výsledky.

Rychlost vyhodnocení neuronových sítí byla měřena na základě vyhodnocení 1000 diabolek. Z množiny zaznamenaných hodnot byly odstraněny extrémy, které vznikly vlivem inicializace grafické karty a zbytek zprůměrován. Změřený výkon lze vidět v tabulce 7.3

Kapitola 8

Závěr

Cílem této diplomové práce bylo prostudovat problematiku měření a kontroly povrchů objektů, navrhnout řešení pro případ výrobku diabolo a výsledné řešení implementovat. Stanovený cíl práce byl zcela splněn.

Po prostudování současných řešení v oboru, jak detekce defektů na povrchu výrobků, tak přesného měření, bylo navrženo řešení skládající se ze dvou částí. První část provádějící měření délky a průměru hlavičky diabolky a druhá část pro detekci defektů pomocí neuronových sítí.

V případě měření délky diabolky je využito prokládání hrany extrahované z gradientního obrazu. Při tomto měření je dosaženo přesnosti s odchylkou do 25 µm, což je o polovinu vyšší přesnost, než bylo kladeno za cíl ve specifikaci. Měření průměru hlavičky se provádí taktéž s pomocí prokládání gradientu polynomy pro sub-pixelovou přesnost. Výsledkem je přesnost tohoto modulu s odchylkou do 10 µm. Na celkové přesnosti měřících algoritmů se podepisuje fakt, že pořízení obrazu probíhá za pohybu diabolky.

Detekce defektů je realizována pomocí neuronových sítí. Byly vyzkoušeny tři přístupy k učení pro dva zvolené modely sítí – Inception v1 a VGG-16. Byly vytvořeny vlastní modely pro pohled na hlavičku a pro pohled zboku. Oba přístupy vyžadují vlastní přístup kvůli různým podmínkám a výskytu vad. I přes rozdílné výkony obou neuronových v části učení, je jejich přesnost shodná. Pro obě sítě bylo dosaženo klasifikační přesnosti 100 %. Dosažení takového výsledku znamená, že modely neuronových sítí, které jsou tvořeny pro řešení mnohem složitějších problému, vykazují velmi dobré výsledky v případě, že datové sady jsou dostatečně disjunktní a prostředí dostatečně stabilní.

Důležitým přínosem této práce je prezentace postupů pro měření a detekci defektů diabolek s aplikací a funkčností v reálném prostředí při výrobě.

V budoucnu by bylo dobré prozkoumat, jak by bylo možné provést korekci diabolky při jejím měření tak, aby se do měření nepromítaly chyby zavedené náklonem diabolky a jestli tento problém neřeší jen samotná kalibrace stroje. Pro měření délky se nyní nepoužívá sub-pixelové prokládání ve fázi hledání hrany. Tento modul by mohl být vylepšen přidáním logiky, která by detekovala profil hrany a ta se tak mohla ideálně proložit. Pro část neuronových sítí by se dalo experimentovat, jak moc se dá daná síť zjednodušovat tak, aby vykazovala stejné výsledky pro tento specifický případ hledání defektů na povrchu diabolky.

Literatura

[1] Abadi, M.; Agarwal, A.; Barham, P.; aj.: TensorFlow: Large-scale machine learning on heterogeneous systems. 2015, software available from tensorflow.org.
URL https://www.tensorflow.org/

[2] Albregtsen, F.; aj.: Statistical texture measures computed from gray level coocurrence matrices. *Image processing laboratory, department of informatics, university of oslo*, ročník 5, 2008.

[3] Altman, D.: *Statistics with confidence: confidence intervals and statistical guidelines*, kapitola Means and their differences. London, United Kingdom: BMJ Books, 2000, ISBN 978-0-727-91375-3, s. 28–36.

[4] Bin, T.; Lei, A.; Jiwen, C.; aj.: Subpixel edge location based on orthogonal Fourier–Mellin moments. *Image and Vision Computing*, ročník 26, č. 4, 2008: s. 563–569.

[5] Blasco, J.; Aleixos, N.; Molto, E.: Computer vision detection of peel defects in citrus by means of a region oriented segmentation algorithm. *Journal of Food Engineering*, ročník 81, č. 3, 2007: s. 535–543.

[6] Bradski, G.: The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[7] Bullock, R.: Least-squares circle fit. 2006.
URL https://dtcenter.org/met/users/docs/write_ups/circle_fit.pdf

[8] Chen, H.; Pang, Y.; Hu, Q.; aj.: Solar cell surface defect inspection based on multispectral convolutional neural network. *Journal of Intelligent Manufacturing*, 2018: s. 1–16.

[9] Chen, Y.; Wang, R.-s.: A method for texture classification by integrating Gabor filters and ICA. *Acta Electronica Sinica*, ročník 35, č. 2, 2007: str. 299.

[10] Chiu, S.-H.; Chou, S.; Liaw, J.-J.; aj.: Textural defect segmentation using a Fourier-domain maximum likelihood estimation method. *Textile research journal*, ročník 72, č. 3, 2002: s. 253–258.

[11] History of C++. Dec 2011.
URL https://en.cppreference.com/w/cpp/language/history

[12] Cubero, S.; Aleixos, N.; Moltó, E.; aj.: Advances in machine vision applications for automatic inspection and quality evaluation of fruits and vegetables. *Food and bioprocess technology*, ročník 4, č. 4, 2011: s. 487–504.

[13] Demant, C.; Garnica, C.; Streicher-Abel, B.: *Industrial image processing: visual quality control in manufacturing*, kapitola Gauging. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ISBN 978-3-642-33905-9, s. 173–201.
URL https://doi.org/10.1007/978-3-642-33905-9_7

[14] Elbehiery, H.; Hefnawy, A.; Elewa, M.: Surface defects detection for ceramic tiles using image processing and morphological techniques. 2005.

[15] Faghih-Roohi, S.; Hajizadeh, S.; Núñez, A.; aj.: Deep convolutional neural networks for detection of rail surface defects. In *IJCNN*, 2016, s. 2584–2589.

[16] Fisher, R.: Fourier Transform. 2000.
URL https://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm

[17] Fitzgibbon, A.; Pilu, M.; Fisher, R. B.: Direct least square fitting of ellipses. *IEEE Transactions on pattern analysis and machine intelligence*, ročník 21, č. 5, 1999: s. 476–480.

[18] Gabor, D.: Theory of communication. Part 1: The analysis of information. *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering*, ročník 93, č. 26, 1946: s. 429–441.

[19] Gao, C.; Zhou, J.; Zhang, W.-h.: Edge detection based on the newton interpolation's fractional differentiation. *Int. Arab J. Inf. Technol.*, ročník 11, č. 3, 2014: s. 223–228.

[20] Gonzalez, R.: *Digital image processing*, kapitola Digital Image Fundamentals. Upper Saddle River, N.J: Prentice Hall, 2008, ISBN 9780131687288, s. 34–71.

[21] Guan, S.: Strip steel defect detection based on saliency map construction using Gaussian pyramid decomposition. *ISIJ International*, ročník 55, č. 9, 2015: s. 1950–1955.

[22] Han, Y.; Shi, P.: An adaptive level-selecting wavelet transform for texture defect detection. *Image and Vision Computing*, ročník 25, č. 8, 2007: s. 1239–1248.

[23] Hashmi, N.: Why Python as programming language? past, present & Future. Aug 2015.
URL https://www.vizteams.com/blog/why-python-as-programming-language-past-present-future/

[24] Heidenhain (editor): *Dotykové sondy pro obráběcí stroje*, kapitola Princip funkce. Heidenhain, 2018, str. 12.
URL https://www.heidenhain.cz/fileadmin/pdb/media/img/1113984-C1_Dotykové_sondy.pdf

[25] Hermosilla, T.; Bermejo, E.; Balaguer, A.; aj.: Non-linear fourth-order image interpolation for subpixel edge detection and localization. *Image and vision computing*, ročník 26, č. 9, 2008: s. 1240–1248.

[26] Hochreiter, S.; Schmidhuber, J.: Long short-term memory. *Neural computation*, ročník 9, č. 8, 1997: s. 1735–1780.

[27] Jia, H.; Murphey, Y. L.; Shi, J.; aj.: An intelligent real-time vision system for surface defect detection. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, ročník 3, IEEE, 2004, s. 239–242.

[28] Kaehler, A.: *Learning OpenCV 3 : computer vision in C++ with the OpenCV library*. Sebastopol, CA: O'Reilly Media, 2016, ISBN 9781491937990.

[29] Karpathy, A.: CS231n Convolutional neural networks for visual recognition. 2018. URL http://cs231n.github.io/convolutional-networks/

[30] Kiefer, J.; Wolfowitz, J.; aj.: Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, ročník 23, č. 3, 1952: s. 462–466.

[31] Kim, S.; Kim, W.; Noh, Y.-K.; aj.: Transfer learning for automated optical inspection. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, IEEE, 2017, s. 2517–2524.

[32] Kleynen, O.; Leemans, V.; Destain, M.-F.: Development of a multi-spectral vision system for the detection of defects on apples. *Journal of food engineering*, ročník 69, č. 1, 2005: s. 41–49.

[33] Kumar, A.; Pang, G. K.: Defect detection in textured materials using Gabor filters. *IEEE Transactions on industry applications*, ročník 38, č. 2, 2002: s. 425–440.

[34] Lambers, J.: Orthogonalization and Least Squares. In *Numerical Linear Algebra*, The University of Southern Mississippi, 2017, s. 136–150.
URL https://www.math.usm.edu/lambers/mat610/book610.pdf

[35] Latif-Amet, A.; Ertüzün, A.; Erçil, A.: An efficient method for texture defect detection: sub-band domain co-occurrence matrices. *Image and Vision computing*, ročník 18, č. 6-7, 2000: s. 543–553.

[36] Li, P.; Liang, J.; Shen, X.; aj.: Textile fabric defect detection based on low-rank representation. *Multimedia Tools and Applications*, 2017: s. 1–26.

[37] Li, Q.; Wang, M.; Gu, W.: Computer vision based system for apple surface defect detection. *Computers and electronics in agriculture*, ročník 36, č. 2-3, 2002: s. 215–223.

[38] Lin, H.; Li, B.; Wang, X.; aj.: Automated defect inspection of LED chip using deep convolutional neural network. *Journal of Intelligent Manufacturing*, 2018: s. 1–10.

[39] Machuca, R.; Gilbert, A. L.: Finding edges in noisy scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník PAMI-3, č. 1, 1981: s. 103–111.

[40] Malamas, E. N.; Petrakis, E. G.; Zervakis, M.; aj.: A survey on industrial vision systems, applications and tools. *Image and vision computing*, ročník 21, č. 2, 2003: s. 171–188.

[41] Mallat, S. G.: A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 11, č. 7, July 1989: s. 674–693, ISSN 0162-8828.

[42] Monadjemi, A.; Mirmehdi, M.; Thomas, B.: Restructured eigenfilter matching for novelty detection in random textures. *learning*, ročník 5, 2004: str. 13.

[43] Moravské, P.: Úvod do techniky CCD čipů. 2018.
URL https://www.gxccd.com/art?id=303&lang=405

[44] Nalwa, V. S.; Binford, T. O.: On detecting edges. *IEEE transactions on pattern analysis and machine intelligence*, ročník PAMI-8, č. 6, 1986: s. 699–714.

[45] Peng, G.; Zhang, Z.; Li, W.: Computer vision algorithm for measurement and inspection of O-rings. *Measurement*, ročník 94, 2016: s. 828–836.

[46] Pratt, W.: *Digital image processing : PIKS inside*, kapitola Continuous Image Mathematical Characterization. New York: Wiley, 2001, ISBN 0-471-22132-5, s. 3–22.

[47] Radhakrishnan, V.; Mohamed, A.: Neural networks for the identification and control of blast furnace hot metal quality. *Journal of process control*, ročník 10, č. 6, 2000: s. 509–524.

[48] Soh, L.-K.; Tsatsoulis, C.: Texture analysis of SAR sea ice imagery using gray level co-occurrence matrices. *CSE Journal Articles*, 1999: str. 47.

[49] Steger, C.: Subpixel-precise extraction of lines and edges. *International Archives of Photogrammetry and Remote Sensing*, ročník 33, č. 3, 2000: s. 141–156.

[50] Tabatabai, A. J.; Mitchell, O. R.: Edge location to subpixel values in digital imagery. *IEEE transactions on pattern analysis and machine intelligence*, ročník PAMI-6, č. 2, 1984: s. 188–201.

[51] Taylor, S.: CCD and CMOS Imaging Array Technologies: Technology Review. Technická zpráva, Microsoft, May 1998.
URL https://www.microsoft.com/en-us/research/publication/ccd-and-cmos-imaging-array-technologies-technology-review/

[52] Tsai, D.-M.; Chiang, C.-H.: Automatic band selection for wavelet reconstruction in the application of defect detection. *Image and Vision Computing*, ročník 21, č. 5, 2003: s. 413–431.

[53] Vacharanukul, K.; Mekid, S.: In-process dimensional inspection sensors. *Measurement*, ročník 38, č. 3, 2005: s. 204–218.

[54] Čepová, L.; Petřkovská, L.: *Legislativa ve strojírenské metrologii a přesné měření 3D ploch: studijní opora*, kapitola Přehled měřících systémů. Vysoká škola báňská - Technická univerzita Ostrava, 2011, ISBN 978-80-248-2514-4, s. 76–102.

[55] Wang, P.; Krishnan, S. M.; Kugean, C.; aj.: Classification of endoscopic images based on texture and neural network. In *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*, ročník 4, IEEE, 2001, s. 3691–3695.

[56] Wang, T.; Chen, Y.; Qiao, M.; aj.: A fast and robust convolutional neural network-based defect detection model in product quality control. *International Journal of Advanced Manufacturing Technology*, ročník 94, č. 9-12, 2018: s. 3465–3471.

[57] Yamashita, R.; Nishio, M.; Do, R. K. G.; aj.: Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 2018: s. 1–19.

[58] Yang, X.; Pang, G.; Yung, N.: Discriminative training approaches to fabric defect classification based on wavelet transform. *Pattern Recognition*, ročník 37, č. 5, 2004: s. 889–899.

[59] Yang, X. Z.; Pang, G. K.; Yung, N. H. C.: Discriminative fabric defect detection using adaptive wavelets. *Optical Engineering*, ročník 41, č. 12, 2002: s. 3116–3127.

[60] Ye, J.; Fu, G.; Poudel, U. P.: High-accuracy edge detection with blurred edge model. *Image and Vision Computing*, ročník 23, č. 5, 2005: s. 453–467.

[61] Young, I.; Gerbrands, J.; van Vliet, L.: *Fundamentals of image processing*, kapitola Digital Image Definitions. Delft: TU Delft, Faculty of Applied Physics, Pattern Recognition Group, 1995, ISBN 90-75691-01-7, s. 2–6.

[62] Zhao, Y. J.; Yan, Y. H.; Song, K. C.: Vision-based automatic detection of steel surface defects in the cold rolling process: considering the influence of industrial liquids and surface textures. *The International Journal of Advanced Manufacturing Technology*, ročník 90, č. 5-8, 2017: s. 1665–1678.

[63] Zhuang, F.; Yanzheng, Z.; Yang, L.; aj.: Solar cell crack inspection by image processing. In *Business of Electronic Product Reliability and Liability, 2004 International Conference on*, IEEE, 2004, s. 77–80.