



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

POČÍTAČOVÉ VIDĚNÍ PRO SLEDOVÁNÍ 3D TISKU

COMPUTER VISION FOR MONITORING OF 3D PRINTING

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

VILIAM CHUDÁČIK

VEDOUcí PRÁCE

SUPERVISOR

doc. RNDr. PAVEL SMRŽ, Ph.D.

BRNO 2025

Zadání bakalářské práce



160547

Ústav: Ústav počítačové grafiky a multimédií (UPGM)
Student: **Chudáčik Viliam**
Program: Informační technologie
Název: **Počítačové vidění pro sledování 3D tisku**
Kategorie: Počítačové vidění
Akademický rok: 2024/25

Zadání:

1. Seznamte se se způsoby rozpoznávání stavu činnosti strojů na základě vizuální informace.
2. Zpracujte přehled hlavních problémů, které nastávají při tisku na běžných 3D tiskárnách.
3. Na základě získaných poznatků navrhnete a implementujete systém, který dokáže rozpoznat nejčastější případy chyb při 3D tisku.
4. Zpřístupněte vytvořený systém pomocí webového rozhraní tak, aby jej bylo možné použít např. v podmínkách naší fakulty.
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- dle doporučení vedoucího

Při obhajobě semestrální části projektu je požadováno:

- funkční prototyp řešení

Podrobné závazné pokyny pro vypracování práce viz <https://www.fit.vut.cz/study/theses/>

Vedoucí práce: **Smrž Pavel, doc. RNDr., Ph.D.**

Vedoucí ústavu: Černocký Jan, prof. Dr. Ing.

Datum zadání: 1.11.2024

Termín pro odevzdání: 14.5.2025

Datum schválení: 12.11.2024

Abstrakt

Tato práce se zabývá detekcí chyb při 3D tisku pomocí počítačového vidění. Cílem je vytvořit systém schopný v reálném čase rozpoznat chyby, jako je stringování nebo tvorba „špaget“, a upozornit na ně uživatele.

Řešení využívá předtrénovaný model YOLOv11s spuštěný na Raspberry Pi 5 pomocí frameworku MNN a je integrováno do OctoPrintu jako zásuvný modul.

Testování potvrdilo schopnost detekce na úrovni srovnatelné s existujícími systémy, a to i přes mírně vyšší počet falešných detekcí. Výsledky ukazují, že i s omezeným hardwarem lze efektivně detekovat chyby tisku v reálném čase.

Abstract

This thesis focuses on error detection in 3D printing using computer vision. The goal is to develop a system capable of recognizing errors such as stringing or spaghetti formation in real time and notifying the user.

The solution uses a pre-trained YOLOv11s model running on a Raspberry Pi 5 via the MNN framework, integrated as a plugin in OctoPrint.

Testing confirmed detection performance comparable to existing systems, despite a slightly higher number of false positives. Results show that reliable real-time error detection is achievable even with limited computational resources.

Klíčová slova

3D tisk, počítačové vidění, detekce chyb, YOLOv11, Raspberry Pi, MNN, OctoPrint, sledování tisku.

Keywords

3D printing, computer vision, error detection, YOLOv11, Raspberry Pi, MNN, OctoPrint, print monitoring.

Citace

CHUDÁČIK, Viliam. *Počítačové vidění pro sledování 3D tisku*. Brno, 2025. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Pavel Smrž, Ph.D.

Počítačové vidění pro sledování 3D tisku

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. RNDr. Pavla Smrže, Ph.D. Další informace o 3D tisku mi poskytl pan Zdeněk Juříček. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Viliam Chudáčík
11. května 2025

Poděkování

Děkuji vedoucímu bakalářské práce doc. RNDr. Pavlu Smržovi, Ph.D., za cenné rady, připomínky a odborné vedení během zpracování této práce.

Poděkování patří také Zdeňku Juříčkovi za poskytnuté informace o 3D tisku a systémech využívaných na fakultě, které výrazně pomohly při návrhu a testování řešení.

Obsah

1	Úvod	3
2	3D tisk	5
2.1	FDM Tiskárna	5
2.2	Příprava tisku	6
2.3	Problémy vyskytující se při tisku	7
2.4	OctoPrint	10
3	Počítačové vidění	11
3.1	Strojové učení	11
3.2	Neuronové sítě	12
3.3	Úlohy počítačového vidění	13
3.4	Trénování modelu počítačového vidění	14
3.5	Využití počítačového vidění při 3D tisku	17
3.6	Existující řešení pro detekci chyb při 3D tisku	17
4	Návrh řešení	20
4.1	Popis navrhovaného systému	20
4.2	Architektura systému	21
4.3	Příprava datové sady	23
4.4	Trénování modelu	26
5	Implementace	30
5.1	Zásuvný modul	30
5.2	Testování modelu a uveřejnění zásuvného modulu	33
5.3	Serverová část pro webovou aplikaci	35
5.4	Uživatelské rozhraní pro webovou aplikaci	36
6	Testování zásuvného modulu	37
6.1	Testování na simulovaných chybách	37
6.2	Testování na tisku bez chyby	40
7	Závěr	41
	Literatura	42
A	Plakát	44

Seznam obrázků

2.1	FDM tiskárna s popisy jednotlivých částí. Převzato z knihy [14]	6
2.2	Příklad chyby špagety. Převzato z fóra Prusa3D [5].	8
2.3	Lehké stringování na vytisknuté lodičce. Převzato z příspěvku na webovém fóru reddit [15].	8
2.4	Silné stringování na vytisknutém modelu.	9
2.5	Ukázka správně vytisknuté vrstvy (vlevo) a vrstvy s hlavou příliš vysoko nad podložkou, způsobující mezery mezi linkami (vpravo).	9
3.1	Diagram perceptronu. Vytvořeno na základě obrázku v knize [9]	13
3.2	Porovnání jednotlivých úloh počítačového vidění a jejich výsledků [3]. . . .	14
4.1	Obrázek popisující architekturu	21
4.2	Návrh uživatelského rozhraní volitelné webové aplikace vytvořený pomocí nástroje Figma.	23
5.1	Postraní panel v OctoPrintu	31
5.2	Část stránky s nastavením pro webkameru a discord upozornění pro zásuvný modul	32
5.3	Upozornění odeslané na platformu Discord	34
5.4	Upozornění s pauzou odeslané na platformu Telegram	34
5.5	Zobrazení seznamu tiskáren	36
A.1	Plakát	44

Kapitola 1

Úvod

3D tisk, který je typem aditivní výroby, se v posledních letech stal klíčovým nástrojem v širokém spektru průmyslových odvětví. Jeho využití se rozprostírá od prototypování výrobků, přes výrobu protéz a dalších zdravotnických pomůcek, až po projekty v oblasti stavebnictví, kde je využíván pro tisk budov a stavebních prvků. Technologie 3D tisku, která poprvé přišla na scénu v druhé polovině 20. století, se od svého vzniku značně vyvinula. Dnes existuje mnoho typů 3D tiskáren, které pracují na různých principech a s různými materiály, což umožňuje širokou variabilitu v jejich použití. Snižující se cena 3D tiskáren a jejich rostoucí dostupnost vedly k rychlému rozšíření této technologie nejen v průmyslovém sektoru, ale i v domácnostech, kde je často využívána pro hobby účely, jako je tisk modelů, prototypů nebo oprav předmětů používaných v domácnosti.

I přes své výhody a rozmanitá využití čelí 3D tisk stále řadě výzev, přičemž jedním z největších problémů je udržení kvality tisku. Během tiskového procesu může dojít k různým chybám, které mohou vést ke znehodnocení celého tisku. Takovéto problémy nejenže zvyšují náklady na materiál, ale mohou také výrazně prodloužit dobu výroby. Řadu chyb může uživatel tiskárny rozpoznat a opravit, nepřetržité sledování tisku je obtížné, neboť tiskové procesy mohou trvat několik hodin, ale i několik dní.

V tomto kontextu se nabízí využití počítačového vidění, což je technologie, která umožňuje počítačům „vidět“ a interpretovat vizuální informace, podobně jako lidé. Počítačové vidění se již osvědčilo v mnoha oblastech, například pro čtení ručně psaného textu, v autonomních vozidlech, při analýze lékařských snímků nebo v bezpečnostních systémech na letištích. V oblasti 3D tisku se počítačové vidění může uplatnit při monitorování tisku v reálném čase a detekci chyb již v průběhu samotného procesu, čímž se minimalizuje riziko vadného tisku.

Tato práce se zaměřuje na návrh a vývoj aplikace, která využívá počítačového vidění k monitorování a vyhodnocování stavu tisknutého objektu v reálném čase. Aplikace bude schopna identifikovat nejčastější chyby při tisku. Na základě nastavení uživatele aplikace upozorní na možnou chybu a v případě potřeby pozastaví tisk, což uživateli umožní provést kontrolu tisku a rozhodnout se, zda v tiskovém procesu pokračovat, nebo ho ukončit. Součástí práce je také vytvoření webové aplikace, která bude umožňovat sledovat průběh tisků z více tiskáren najednou, včetně aktuálního vyhodnocení tisku na základě počítačového vidění.

V práci jsou v kapitole 2 představeny 3D tiskárny a tisk samotný, spolu s problémy, které se při tisku běžně vyskytují společně s jejich příčinami a jejich řešením. Na konci této kapitoly je také popsán systém OctoPrint, sloužící pro ovládání tiskáren.

Následující kapitola 3 popisuje neuronové sítě, počítačové vidění a jednotlivé úlohy počítačového vidění s výběrem vhodných úloh pro tuto práci. Součástí je také představení existujících řešení detekce chyb při tisku pomocí počítačového vidění a zhodnocení možnosti integrací s tiskárnou.

Návrh architektury systému detekce chyb a uživatelského rozhraní pro zobrazení stavu tiskáren, nastavení možností detekce a upozornění se pak nachází v kapitole 4. Tato kapitola také obsahuje informace o způsobu trénování modelu, srovnání výsledků jednotlivých modelů a výběr modelu pro implementaci.

Implementace uživatelského rozhraní a systému pro detekci s upozorněním je popsána v kapitole 5. Tato kapitola popisuje způsob implementace zásuvného modulu pro OctoPrint a jeho modulu pro počítačové vidění. Dále se zde nachází popis implementace volitelné webové aplikace. Kapitola 6 pak popisuje způsob testování zásuvného modulu na skutečných tiscích a popisuje výsledky jednotlivých testů.

Kapitola 2

3D tisk

3D tiskárny pro výrobu využívají procesu aditivní výroby. Tento proces spočívá v postupném a kontrolovaném přidávání materiálu, čímž postupně vzniká požadovaný výrobek. Tento princip se liší od tradičních subtraktivních metod, jako je soustružení, kde se materiál naopak odebírá. Dalšími příklady aditivní výroby jsou například odlévání a nebo spékání.

První komerčně dostupná 3D tiskárna přišla koncem 80. let 20. století a byla nazvaná SLA-1. Tiskárna fungovala na principu vytvrzování tekutého fotopolymeru pomocí UV laseru. Moderní SLA tiskárny dnes místo laseru často využívají UV displeje s vysokým rozlišením, což umožňuje větší přesnost a efektivitu tisku. Sledování chyb u SLA tiskáren pomocí kamery je však složitější, jelikož tyto tiskárny jsou obvykle uzavřeny v ochranném oranžovém krytu nebo skleněné komoře, která blokuje UV záření.

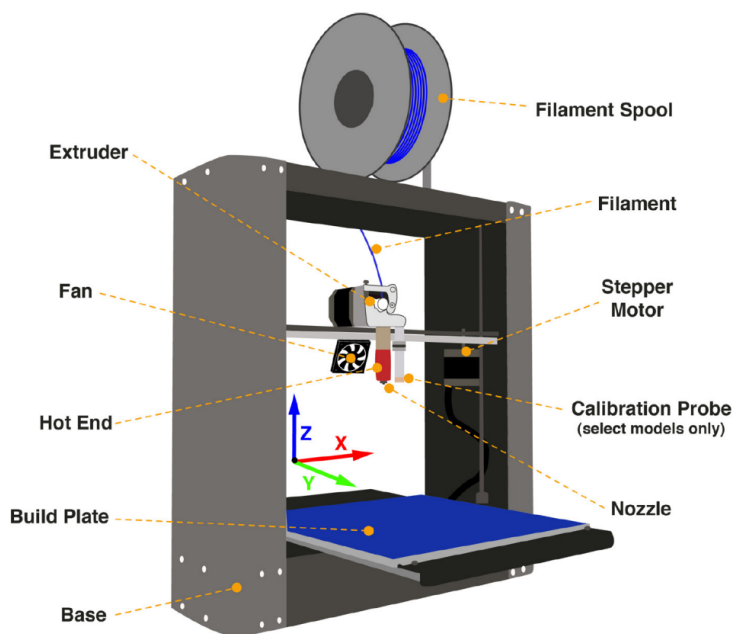
Mezi nejrozšířenější a nejdostupnější technologie pro domácí i průmyslové využití patří tiskárny typu FDM (Fused Deposition Modeling). První FDM tiskárnu uvedla na trh v roce 1992 společnost Stratasys. FDM tiskárny využívají technologie modelování depozicí taveniny, fungují na principu tavení materiálu (nejčastěji plastové struny) v extruderu neboli tiskové hlavě při teplotách okolo 200 °C. Tisková hlava následně nanáší tavený materiál ve vrstvách na tiskovou podložku. Průběh tisku v těchto tiskárnách je jednodušší sledovat, jelikož tyto tiskárny jsou většinou větší a otevřené. Hlavním zdrojem pro tuto kapitolu je [14] a [10].

2.1 FDM Tiskárna

FDM tiskárna se skládá z několika klíčových částí, které společně zajišťují přesný pohyb, tavení a ukládání materiálu. Její konstrukce bývá poměrně modulární a některé komponenty se mohou lišit v závislosti na konkrétním modelu, nicméně většina tiskáren sdílí tyto základní části:

- **Rám a základna (anglicky *Frame and base*) tiskárny** – nosná konstrukce, která drží všechny ostatní komponenty. Může být tvořena z hliníkových profilů, plastových dílů nebo plechových rámců. Stabilita rámu má zásadní vliv na kvalitu tisku.
- **Tisková podložka (anglicky *Bed* nebo *Build plate*)** – povrch, na který je ukládán tisknutý objekt. Podložka bývá vyhřívaná, což zlepšuje přilnavost materiálu a zabráňuje deformacím. Povrch podložky bývá pokryt různými materiály (sklo, PEI).
- **Krokové motory (anglicky *Stepper motor*)** – zajišťují pohyb tiskové hlavy a podložky pomocí řemenů nebo trapézových šroubů.

- **Tisková hlava**
 - **Extruder** – zařízení, které podává filament do trysky.
 - **Hotend a tryska (anglicky *Nozzle*)** – v hotendu je filament zahříván na tiskovou teplotu a následně protlačován tryskou. Trysky mají různé průměry (nejčastěji 0,4 mm). Hotend obsahuje topné těleso, termistor a chladič blok s ventilátorem.
 - **Chlazení** – aktivní chlazení je důležité pro správné ochlazení vytlačeného plastu, čímž se zlepšuje kvalita tisku.
 - **Kalibrační sonda (anglicky *Calibration probe*)** – zařízení sloužící k měření a kalibraci vzdálenosti mezi tryskou a podložkou. Tato sonda se nemusí nacházet na všech tiskárnách.
- **Cívka s filamentem (anglicky *Filament spool*)** – Na cívce je navinut materiál ve formě struny, obvykle o průměru 1,75 mm. Filament se před tiskem zavádí do extruderu.
- **Displej a ovládací panel** – umožňuje uživateli spouštět tisk, kalibrovat tiskárnu a zobrazovat informace o průběhu tisku.



Obrázek 2.1: FDM tiskárna s popisy jednotlivých částí. Převzato z knihy [14]

2.2 Příprava tisku

Proces 3D tisku začíná vytvořením nebo získáním 3D modelu. Model lze vytvořit v CAD softwaru, který slouží k počítačem podporovanému navrhování a modelování objektů. CAD aplikace, jako například Autodesk Inventor, SolidWorks nebo Fusion 360, umožňují nejen

návrh, ale také export modelů ve formátu STL. Tento formát je standardem pro 3D tisk v SLA i FDM tiskárnách a popisuje geometrii objektu pomocí trojúhelníkových ploch.

Alternativou k vlastní tvorbě modelu je jeho stažení z online databází, které nabízejí širokou škálu modelů připravených k tisku. Mezi nejznámější platformy patří Thingiverse¹ a Printables².

Jakmile máme STL soubor, je nutné jej zpracovat v tzv. sliceru – softwaru, který převádí 3D model do GCODE souboru obsahujícího instrukce pro tiskárnu. V sliceru je důležité správně nastavit tiskové parametry, jako jsou výška vrstvy, rychlost tisku, teplota extruderu a podložky. Také je potřeba vybrat profil tiskárny a materiálu, který bude použit.

Mezi nejběžnější tiskové materiály patří PLA, PETG, ABS, TPU a TPE. Každý z těchto materiálů má odlišné vlastnosti, například rozdílnou odolnost vůči teplotním vlivům, pružnost nebo přilnavost k podložce. PLA je nejpoužívanější díky své snadné tisknutelnosti, zatímco ABS je odolnější vůči mechanickému namáhání a vyšším teplotám. TPU a TPE jsou elastické materiály vhodné pro výtisky, které vyžadují pružnost a odolnost vůči nárazům.

Po nastavení všech parametrů lze vygenerovat GCODE soubor, který je nutné přenést do tiskárny. Možnosti přenosu závisí na konkrétním modelu tiskárny – některé podporují přenos přes paměťovou kartu nebo USB disk, jiné umožňují propojení přes USB kabel či bezdrátové připojení pomocí Wi-Fi.

Správná příprava tisku je klíčová pro dosažení kvalitních výsledků, a proto je důležité věnovat dostatečnou pozornost jak tvorbě modelu, tak správnému nastavení sliceru a tiskárny.

2.3 Problémy vyskytující se při tisku

Problémy při 3D tisku mohou nastat z mnoha důvodů, například odlepením výtisku od tiskové podložky, špatnou kalibrací tiskové podložky, chyby v tisknutém modelu, poruch tiskárny nebo špatné nastavení tisku. Zdroji pro tuto kapitolu jsou [1], [18] a [4].

Přeextrudování (anglicky *Overextrusion*)

Přeextrudování se projevuje deformací venkovních stěn tištěného modelu a hromaděním materiálu na horních vrstvách. Tato chyba je způsobená vyšším tokem materiálu, než je potřeba. Řešením je snížení toku materiálu v nastavení sliceru.

Podextrudování (anglicky *Underextrusion*)

Projevem podextrudování jsou naopak mezery mezi jednotlivými linkami plastu a mezi obvody výrobku. Tento problém může být způsoben příliš nízkým průtokem materiálu nebo částečně ucpanou tryskou. Řešením je zkontrolování trysky nebo zvýšení hodnoty průtoku ve sliceru.

Špagety

Špagety jsou jeden z častých problémů při 3D tisku. Příčinou je odlepení tisknutého modelu od tiskové podložky a následné tisknutí materiálu do vzduchu, nebo chyba v tisknutém

¹<https://www.thingiverse.com/>

²<https://www.printables.com/>

modelu, která způsobí tisknutí materiálu do vzduchu. Tento problém se může vyskytnout kdykoli během tisku a může způsobit nemalé ztráty materiálu, pokud tato chyba není odhalena včas.



Obrázek 2.2: Příklad chyby špagety. Převzato z fóra Prusa3D [5].

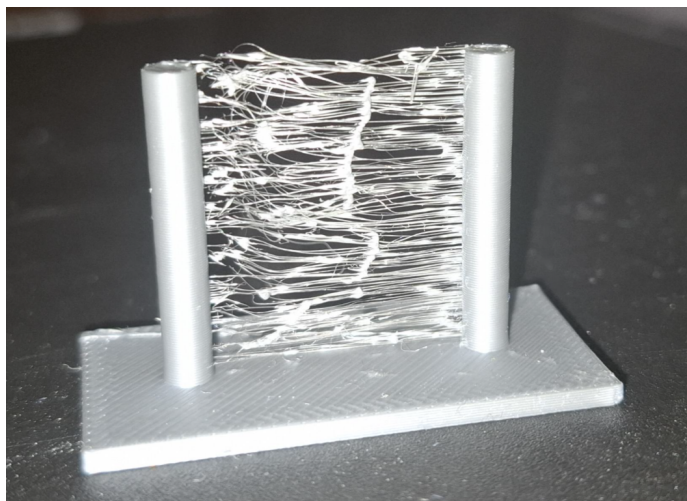
Stringování

Stringování se projevuje tím, že tryska při pohybu zanechává tenká vlákna plastu. Příčinou může být nastavení tisku nebo vlastnosti využitého materiálu. Tato chyba se projevuje v průběhu celého tisku. Lehký projev této chyby bývá relativně jednoduché opravit po dotisknutí výrobku například krátkým nahřátím výrobku horkovzdušnou pistolí.

Předcházet tomuto problému je možné například změnou parametrů pro retrakci, resp. délkou a rychlostí zpětného pohybu filamentu v extruderu při pohybu tiskové hlavy, nebo snížením teploty hotendu.



Obrázek 2.3: Lehké stringování na vytisknuté lodičce. Převzato z příspěvku na webovém fóru reddit [15].



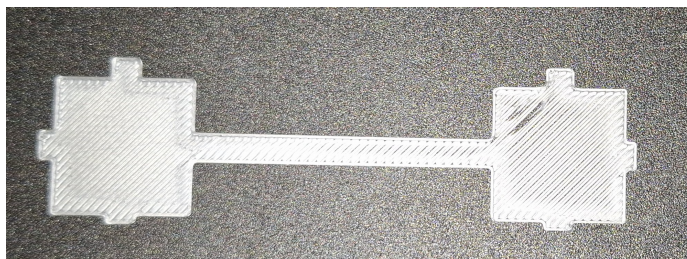
Obrázek 2.4: Silné stringování na vytisknutém modelu.

Špatně vytisknutá první vrstva

Problém s vytisknutím první vrstvy se může projevit různými způsoby v závislosti na přesné příčině. Pokud je špatně kalibrovaná tisková podložka a tisková hlava je příliš nízko, může dojít k tomu, že se na tiskovou podložku nevytlačí žádný materiál, nebo že vytlačený materiál při tisku teče více do stran a vytváří tak první vrstvu s velmi hrubou texturou. Problém nastává i v případě, že je tisková hlava naopak příliš vysoko, to se může projevit odlepením materiálu od tiskové podložky nebo tím, že se materiál při tisku neslévá a zanechává tak mezery v linkách tiskuté vrstvy.

Tento problém lze řešit správnou kalibrací výšky první vrstvy (Označováno jako Z-offset). Některé tiskárny umožňují kalibraci automatickou, některé pouze manuální pomocí kalibračních šroubů.

Další příčinou pro odlepení první vrstvy mohou být také nečistoty na tiskové podložce.



Obrázek 2.5: Ukázka správně vytisknuté vrstvy (vlevo) a vrstvy s hlavou příliš vysoko nad podložkou, způsobující mezery mezi linkami (vpravo).

Materiál přestal být vytlačován během tisku

Pokud se tisková hlava běžně pohybuje jako při tisku, ale nedochází k vytlačování materiálu, je možné, že došel tiskový materiál, tisková struna praskla nebo došlo k ucpání tiskové hlavy. Zde je nejprve potřeba zkontrolovat, zda máme dost tiskového materiálu a je správně zasunutý v tiskové hlavě a ta má správně nastavenou teplotu. Pokud je vše správně a ma-

terál stále nevychází z tiskové hlavy, je potřeba postupovat jako v případě ucpané tiskové hlavy.

Ucpaná tisková hlava

Při tisku může dojít také k ucpaní tiskové hlavy nebo trysky, a to při změně materiálů, zejména pokud přecházíme z materiálu vyžadujícího vyšší tiskovou teplotu na materiál s nižší teplotou. K tomuto problému může dojít jak před, tak během tisku.

Například při přechodu z PLA na ABS může v trysce zůstat malé množství PLA, které je následně vystaveno teplotám vyšším, než je jeho běžný bod tavení. V důsledku toho se může PLA přepálit, což může vést k částečnému nebo úplnému ucpaní tiskové hlavy.

Dalšími důvody ucpaní mohou být nečistoty, které se do tiskové hlavy dostanou spolu s materiálem.

Pro řešení tohoto problému je důležité konzultovat příručku tiskárny, jelikož doporučené postupy se mohou lišit. Pro začátek je možné zkusit uvolnit trysku například pomocí špendlíku.

2.4 OctoPrint

OctoPrint je aplikace s otevřeným kódem (anglicky open-source), která umožňuje vzdálené ovládání a monitorování 3D tiskáren na lokální síti. Tento software je oblíbený zejména mezi uživateli, kteří chtějí rozšířit funkcionalitu své tiskárny a získat větší kontrolu nad tiskovým procesem. OctoPrint je možné provozovat na různých zařízeních, jako jsou počítače, notebooky nebo mobilní telefony, ale nejčastěji se instaluje na minipočítače Raspberry Pi. Důvodem jsou kompaktní rozměry, nízká cena a spotřeba energie.

Mezi hlavní funkce OctoPrintu patří ovládání tiskárny na dálku, monitorování teploty extruderu a tiskové podložky, nahrávání GCODE souborů a spouštění tiskových úloh. OctoPrint také podporuje připojení kamery, což umožňuje sledování průběhu tisku v reálném čase a tvorbu časosběrných videí.

Jednou z největších výhod OctoPrintu je možnost jeho rozšíření prostřednictvím zásuvných modulů (pluginů), které vyvíjí komunita uživatelů. Díky těmto modulům lze například umožnit vzdálený přístup mimo lokální síť, automatizovat řízení napájení tiskárny a dalších připojených zařízení, integrovat upozornění nebo analyzovat kvalitu tisku pomocí počítačového vidění [17].

Kapitola 3

Počítačové vidění

Počítačové vidění se zabývá vývojem metod pro získávání a interpretaci vizuálních informací z obrazových dat. Zatímco lidé přirozeně vnímají hloubku, tvary a pohyb objektů, naučit stroje tuto schopnost představuje složitý výpočetní problém. I přes výrazný pokrok v posledních desetiletích zůstává plná automatická interpretace obrazu výzvou.

Hlavním důvodem složitosti počítačového vidění je skutečnost, že se jedná o inverzní problém – z dvourozměrného obrazu se snažíme rekonstruovat trojrozměrný svět s neúplnými informacemi. K tomu využíváme fyzikální a pravděpodobnostní modely, stejně jako strojové učení. Inspirací pro algoritmy počítačového vidění jsou jak principy lidského vnímání, tak znalosti z optiky a počítačové grafiky, které popisují šíření světla a jeho interakci s objekty.

Díky moderním metodám, jako je stereo rekonstrukce nebo hluboké neuronové sítě, je dnes možné vytvářet přesné 3D modely prostředí, rozpoznávat objekty na fotografiích nebo dokonce odhadovat lidské emoce z výrazů tváře. Přesto však stále zůstává rozdíl mezi lidským vnímáním a schopnostmi strojů, což činí z počítačového vidění dynamicky se rozvíjející oblast výzkumu.

V rámci této práce bude počítačové vidění využito pro automatickou detekci chyb během procesu 3D tisku. Klíčovým požadavkem je schopnost rychle a spolehlivě identifikovat typické vady. Výzvou je zejména práce v reálném čase, variabilita tiskových podmínek (například různé materiály, tiskárny, prostředí a osvětlení), nutnost minimalizace falešných poplachů a nízká náročnost na výpočetní výkon. Hlavními zdroji pro tuto kapitolu jsou [16] a [21].

3.1 Strojové učení

Strojové učení je nedílnou součástí počítačového vidění a jeho význam neustále roste. Už od 70. let 20. století bylo spojováno s umělou inteligencí, digitálním zpracováním obrazu a rozpoznáváním vzorů, dnes známým jako strojové učení.

Po dlouhou dobu se počítačové vidění spoléhalo na ručně navržené algoritmy, jejichž parametry byly optimalizovány pomocí tradičních statistických metod. Skutečná revoluce však nastala s rozvojem rozsáhlých datových sad, jako jsou ImageNet nebo COCO, které umožnily efektivní trénování modelů. Díky tomu se staly hluboké neuronové sítě dominantní technologií nejen pro klasifikaci a segmentaci obrazu, ale také pro úlohy, jako je rekonstrukce hloubky, odhad pohybu či zlepšení kvality obrazu.

3.2 Neuronové sítě

Neuronové sítě, známé také jako umělé neuronové sítě, jsou modely strojového učení inspirované strukturou a funkcí lidského mozku. Skládají se z propojených vrstev uzlů (umělých neuronů), které jsou organizovány do vstupní vrstvy, jedné nebo více skrytých vrstev a výstupní vrstvy. Každý uzel je propojen s ostatními a má přiřazenou váhu a práh. Pokud výstup uzlu překročí jeho prahovou hodnotu, aktivuje se a předá data do další vrstvy, jinak zůstane neaktivní.

První model neuronové sítě, perceptron, představil Frank Rosenblatt. Dnes je ale, častěji použití tzv. sigmoidních neuronů, tyto neurony umožňují malé úpravy hodnot těchto neuronů což způsobí pouze malé změny na výstupu, což umožňuje jejich trénování.

Neuronové sítě se učí prostřednictvím tréninkových dat, která pomáhají upravit váhy spojení mezi neurony a tím zlepšují jejich přesnost. Jakmile jsou vyladěny, stávají se výkonnými nástroji umělé inteligence, schopnými úloh, jako je rozpoznávání řeči a obrazu.

Neuronové sítě jsou tedy klíčovou součástí moderního strojového učení a umělé inteligence, protože poskytují efektivní rámec pro modelování složitých vzorců a rozhodování na základě dat. Jedním z druhů neuronových sítí jsou sítě konvoluční.

Konvoluční neuronové sítě

Konvoluční neuronové sítě jsou klíčovým prvkem v oblasti zpracování obrazu a počítačového vidění. Hlavním principem CNN je schopnost učit se a detekovat lokální vzory v obrazech pomocí konvolučních operací, které jsou základem pro výstavbu mapy vlastností (anglicky *feature map*), tedy prostorů, které reprezentují různé aspekty obrazu jako jsou rohy, kraje a textury.

Na rozdíl od tradičních neuronových sítí, kde jsou všechny jednotky v jedné vrstvě propojeny s jednotkami v předchozí vrstvě, konvoluční vrstvy organizují tyto jednotky do mapy vlastností. V konvolučních vrstvách jsou váhové součty prováděny pouze v malém místním okně nebo-li jádru, a váhy jsou stejné pro všechny pixely v daném okně, což je analogické k běžné konvoluci obrazu.

V konvolučních vrstvách jsou aktivace z předchozí vrstvy lineárně kombinovány s použitím různých konvolučních jader pro každý výstupní kanál. Tento přístup umožňuje vytvoření specifických rysů, které jsou poté kombinovány, aby vznikly složitější a semantičtější rysy.

Typy vrstev v konvoluční síti

Konvoluční sítě obsahují několik druhů vrstev, které spolu pracují na extrakci a zpracování informací z vstupních dat. Zdrojem pro tuto část je [22]:

- **Konvoluční vrstva** Tato vrstva provádí konvoluci mezi vstupními daty a konvolučními jádry (filtry). Hlavním úkolem je extrakce lokálních rysů z obrazu.
- **Aktivační funkce** Tato vrstva se využívá po konvoluční vrstvách a slouží k tomu, aby síť byla schopna naučit se komplexnější vztahy mezi daty. Nejčastěji využívanou funkcí je ReLU, která převádí pozitivní vstupní hodnotu přímo, a pokud je vstupní hodnota záporná nebo nulová, výstup je nula.
- **Poolingová vrstva** Slouží ke zmenšení prostorového rozměru výstupu z předchozích vrstev, což vede k snížení výpočetního nákladu. Obvykle se používají operace max

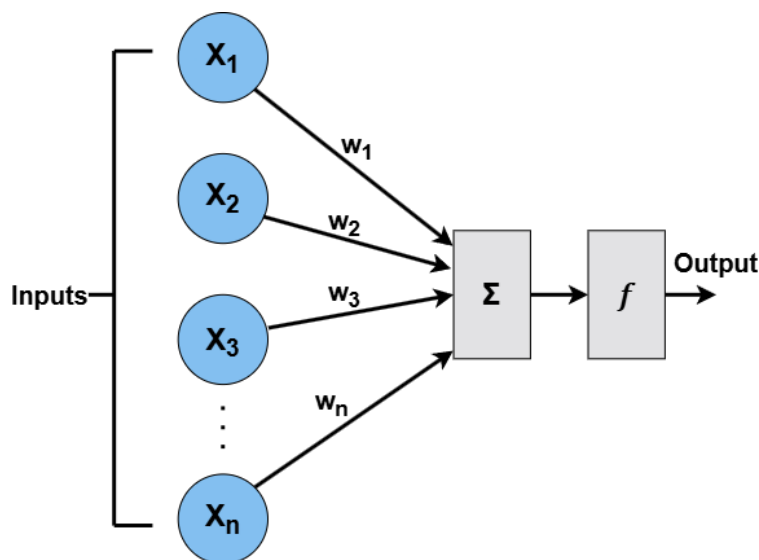
pooling nebo average pooling. Tyto operace na výstupu vracejí buď nejvyšší hodnotu (max pooling), nebo průměrnou hodnotu (average pooling) z daného vstupu.

- **Plně propojené vrstvy** Většinou se nachází na konci sítě a slouží k provádění finální klasifikace nebo predikce.

Na Obrázku 3.1 vidíme diagram perceptronu (umělého neuronu) se vstupy označenými jako $(x_1, x_2, x_3, \dots, x_n)$. Každý z těchto vstupů x_i má svojí přiřazenou váhu w_i , která určuje důležitost tohoto vstupu. Následuje výpočetní část neuronu. Prvním krokem je součtová funkce (značená symbolem Σ), která provádí vážený součet vstupů podle vzorce:

$$z = \sum x_i \cdot w_i + b \text{ (bias)}$$

Výstup z pak vstupuje do aktivační funkce f , která na základě své definice rozhodne, zda má být perceptron aktivován.



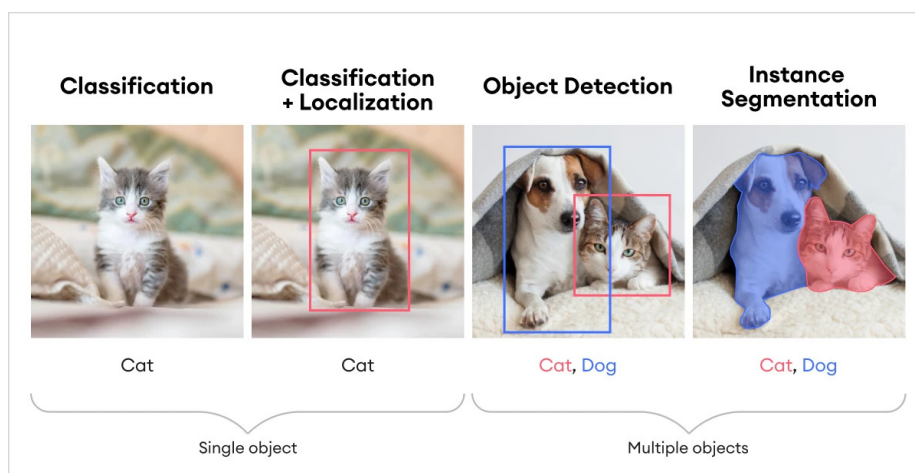
Obrázek 3.1: Diagram perceptronu. Vytvořeno na základě obrázku v knize [9]

3.3 Úlohy počítačového vidění

Počítačové vidění se v praxi zaměřuje na různé úkoly, které se liší mírou složitosti i způsobem zpracování obrazových dat. Mezi základní úlohy patří klasifikace, detekce objektů a segmentace. Zdrojem pro část o segmentaci je [8]

- **Klasifikace obrazu** — Cílem je přiřadit celému obrázku jednu třídu. Typickým příkladem je určení, zda je na snímku pes nebo kočka. Tato úloha předpokládá, že na obrázku je právě jeden hlavní objekt.
- **Detekce objektů** — Kromě určení tříd objektů se model snaží určit i jejich pozici v obraze pomocí ohraňujících rámečků (bounding boxes). Detekce objektů umí narozdíl od klasifikace pracovat i s obrázky obsahující více různých objektů. Typické je použití v oblasti autonomního řízení.

- **Segmentace** — Nejdetailnější úloha, kde se každému pixelu přiřazuje konkrétní třída. Díky tomu lze přesně určit tvary a hranice objektů v obraze. Segmentace nachází využití například v pokročilých asistentech řízení, navigaci robotů, medicínském zpracování obrazu, satelitním snímkování nebo rozšířené realitě. Segmentace může být dále rozdělena do několika podtypů:
 - **Sémantická segmentace** – přiřazuje každému pixelu příslušnost k dané třídě bez rozlišení jednotlivých instancí objektů. Například všechny osoby na obrázku budou označeny jednou barvou.
 - **Segmentace instancí** – navíc rozlišuje jednotlivé objekty stejné třídy (například osoba č. 1, osoba č. 2). Užitečná při sledování více objektů.
 - **Panoptická segmentace** – kombinuje přístup sémantické a instance segmentace a je považována za aktuálně nejkomplexnější typ segmentace.



Obrázek 3.2: Porovnání jednotlivých úloh počítačového vidění a jejich výsledků [3].

Každá z těchto úloh má své využití a klade odlišné nároky na architekturu neuronové sítě i na formát vstupních dat.

Jelikož během 3D tisku může dojít k více chybám současně, například kombinaci stringování a vzniku špaget, není příliš vhodné využívat klasifikaci snímků, která přiřazuje celému snímku pouze jednu kategorii. Vhodnější je proto využití metod detekce objektů nebo segmentace, které umožňují identifikovat více různých chyb na jednom snímku. Detekce objektů představuje kompromis mezi složitostí modelu a detailností výsledku, zatímco segmentace nabízí nejvyšší míru přesnosti na úrovni jednotlivých pixelů, avšak za cenu vyšší výpočetní náročnosti. V rámci této práce budou testovány modely pro detekci objektů i segmentaci a výsledně užitý model bude vybrán na základě dosažených výsledků. Z tohoto důvodu, je nutné, aby vytvořená datová sada obsahovala anotace určené pro segmentaci, které je možné převést na ohraničující rámečky a využít i pro trénování detekce objektů.

3.4 Trénování modelu počítačového vidění

Úspěšné trénování modelu hlubokého učení začíná u kvalitních a dobře připravených dat. Výběr vhodné trénovací, validační a testovací sady je zásadní pro to, aby se model naučil

řešit daný úkol tak, jak bude později vyžadováno v praxi. Přitom je důležité nejen množství dat, ale také to, aby všechna data odpovídala podmínkám, ve kterých bude model skutečně nasazen. Pokud jsou například trénovací data tvořena profesionálními obrázky, ale nasazení modelu má probíhat na běžných fotografiích z mobilního telefonu, hrozí výrazný pokles přesnosti.

Vedle samotné kvality a konzistence dat hraje důležitou roli i volba architektury modelu a nastavení trénovacích parametrů. Ty ovlivňují rychlost učení, stabilitu trénování a schopnost modelu generalizovat. Nicméně ani ten nejlepší model s pečlivě nastavenými parametry nebude fungovat spolehlivě, pokud vstupní data neodráží reálné vstupy, se kterými se model setká v praxi. To se může projevit například tak, že model vykazuje dobré výsledky na validační sadě, ale selhává při testování nebo ostrém nasazení.

Je proto vhodné věnovat dostatečnou pozornost návrhu celé datové sady již od počátku vývoje. Modely hlubokého učení jsou sice schopné zachytit složité vzory ve vstupních datech, ale pouze tehdy, pokud se s těmito vzory během trénování setkají. Ani pokročilá architektura nedokáže kompenzovat nekvalitní nebo irelevantní data.

V neposlední řadě hraje roli i velikost datové sady – obecně platí, že čím větší množství trénovacích dat, tím lépe. Velké objemy dat umožňují modelu zachytit jemné nuance a zvyšují jeho schopnost zobecňovat na nové případy. Zdroji pro tuto část jsou [9] a [19].

Datové sady

Jelikož pro vytvoření kvalitního modelu je třeba velkého množství anotovaných dat, nabízí se využití předtrénovaných modelů počítačového vidění namísto pouhého použití jejich architektury. Tyto modely jsou předtrénovány na velkých datových sadách, jako je ImageNet nebo MS COCO. Tyto datové sady obsahují statisíce až miliony anotovaných snímků s různými objekty nebo zvířaty. Model si na těchto datech osvojí vytváření map vlastností, které mu umožňují rozpoznávat například hrany ve snímcích. Díky těmto naučeným vlastnostem lze výrazně urychlit a zefektivnit trénování modelu na specifický požadavek, zejména v případě použití menších datových sad.

I při práci s většími datovými sadami je však možné využít předtrénované modely a navázat na jejich již naučené vlastnosti. Tento postup označujeme jako přenášené učení (anglicky *transfer learning*) a následné doladění (anglicky *fine-tuning*) modelu. Vzhledem k tomu, že vlastní trénování modelů na rozsáhlých datových sadách je nejen časově, ale i výpočetně velmi náročné, výzkumníci často své předtrénované modely veřejně sdílejí.

Rozdělení datové sady pro trénování

Aby bylo možné model efektivně natrénovat a zároveň objektivně zhodnotit jeho schopnost generalizace, je nutné rozdělit dostupná data do několika částí. Nejčastěji se využívá trojí dělení na trénovací, validační a testovací sadu. Trénovací data slouží k samotnému procesu učení – tedy k aktualizaci vah modelu na základě zpětné vazby z chyb. Validační sada umožňuje průběžně sledovat, jak si model vede mimo trénovací vzorky, a případně upravit trénovací strategii, například změnou hyperparametrů. Porovnání ztrátové funkce na trénovací a validační sadě pak umožňuje odhalit přetrénování modelu.

Způsob rozdělení datové sady závisí na množství dostupných dat. V případě velmi malých datových sad je možné vynechat validační nebo testovací sadu, případně upravit poměry rozdělení ve prospěch trénovací sady. Důležité je také zajistit, aby každá část obsahovala podobný poměr jednotlivých detekovaných kategorií. V opačném případě by mohlo

dojít ke zkreslení výsledků a snížení schopnosti modelu správně generalizovat na všechny třídy dat.

Hodnocení výkonu modelu

Pro správné porozumění evaluačním metrikám je nejprve třeba definovat základní pojmy, na kterých tyto metriky staví.

- **Skutečně pozitivní (True Positive, TP)** – model správně detekoval objekt, který se na snímku skutečně nachází.
- **Falešně pozitivní (False Positive, FP)** – model detekoval objekt, který se na snímku ve skutečnosti nenachází.
- **Falešně negativní (False Negative, FN)** – model přehlédl objekt, který se na snímku nachází.
- **Skutečně negativní (True Negative, TN)** – model správně vyhodnotil, že se na snímku daný objekt nenachází.

Pro posouzení kvality natrénovaného modelu se používají různé evaluační metriky. Každá z nich poskytuje odlišný pohled na výkon modelu a jejich kombinace pomáhá lépe porozumět silným i slabým stránkám systému:

- **IoU** – Měří překryv mezi predikovaným a skutečným objektem. Vypočítává se jako podíl plochy průniku a sjednocení dvou oblastí.

Pro výpočet IoU je potřeba skutečná oblast ($B_{\text{skutečný}}$), kterou jsme vytvořili při anotaci dat. Má formát ohraničujícího rámečku nebo segmentační masky. Dále je potřeba predikovaná oblast ($B_{\text{predikovaný}}$), kterou predikoval trénovaný model. Tyto hodnoty se poté dosadí do následujícího vzorečku:

$$\text{IoU} = \frac{B_{\text{skutečný}} \cap B_{\text{predikovaný}}}{B_{\text{skutečný}} \cup B_{\text{predikovaný}}}$$

Pomocí IoU je možné určit, zda jsou výsledky skutečně pozitivní nebo falešně pozitivní. K tomuto účelu se nastavují prahové hodnoty, nejčastěji 0,5 nebo 0,75. To znamená, že pokud překryv mezi predikovaným a skutečným objektem dosahuje alespoň 50 % (respektive 75 %), výsledek je považován za skutečně pozitivní. V opačném případě se jedná o falešně pozitivní výsledek.

- **Přesnost (Precision)** – Udává, kolik procent z detekovaných objektů bylo validně identifikováno. Pro výpočet je potřeba znát počet skutečně pozitivních hodnot (TP) a počet falešně pozitivních (FP) hodnot:

$$\text{Přesnost} = \frac{TP}{TP + FP}$$

- **Úplnost (Recall)** – Udává, kolik procent z existujících relevantních objektů model úspěšně detekoval. Výpočet se provádí jako podíl počtu skutečně pozitivních výsledků (TP) a součtu skutečně pozitivních (TP) a falešně negativních (FN) výsledků.

$$\text{Úplnost} = \frac{TP}{TP + FN}$$

- **F1 skóre** – Harmonický průměr přesnosti a úplnosti, který vyvažuje obě hodnoty do jednoho čísla. Vhodné zejména při nevyvážených datech. Výpočet se provádí s využitím hodnot přesnosti (p) a úplnosti (r) dle následujícího vzorce:

$$\text{F-score} = \frac{2pr}{p + r}$$

- **PR křivka (Precision-Recall)** – PR křivka graficky znázorňuje vztah mezi přesností a úplností modelu při různých prahových hodnotách rozhodovací jistoty. Pro každou detekovanou třídu lze vygenerovat samostatnou křivku, která ukazuje, jak se mění výkon modelu při uvolňování nebo zpřísňování rozhodovacího prahu.

Kvalitní model si dokáže udržet vysokou přesnost i při rostoucí úplnosti – tedy i při nižších prahových hodnotách. Znamená to, že dokáže detekovat více objektů (vyšší úplnost), aniž by výrazně narůstal počet falešně pozitivních případů (nižší přesnost).

Naproti tomu méně kvalitní model musí pro dosažení vyšší úplnosti „obětovat“ přesnost, protože detekuje i objekty, které se na snímku ve skutečnosti nenacházejí.

- **mean Average Precision (mAP)** – Patří mezi nejčastěji používané metriky pro hodnocení detekce objektů. Hodnota je udávána v procentech a vyjadřuje průměrnou přesnost modelu napříč různými třídami a prahovými hodnotami překryvu (IoU). Výpočet spočívá ve zprůměrování ploch pod PR křivkami všech detekovaných tříd.

3.5 Využití počítačového vidění při 3D tisku

Použití počítačového vidění při sledování procesu 3D tisku je možné díky tomu, že většinu chyb lze vizuálně rozpoznat, a to i na snímcích z běžné kamery. Kamery umístěné v oblasti tiskové plochy umožňují monitorovat průběh tisku a detekovat chyby, jako jsou špagety, preextrudování či deformace. Určitou komplikací však může být tisková hlava, která může zakrýt právě tisknutou oblast. Přesnost a úspěšnost detekce je také ovlivněna kvalitou kamery a osvětlením tiskové plochy. Problémy s detekcí mohou nastat zejména u chyb menšího rozsahu nebo v případě, kdy k vadě dojde na opačné straně modelu, která není v zorném poli kamery. Některá řešení využívají více kamer namířených na tiskovou plochu, což umožňuje pokrytí více úhlů pohledu a zvýšení spolehlivosti detekce. Tato řešení jsou však technicky i finančně náročnější a v rámci této práce nebudou využívána [7]. V následující části jsou popsány příklady řešení detekce chyb pomocí počítačového vidění z komerční sféry i z odborných prací.

3.6 Existující řešení pro detekci chyb při 3D tisku

Detekce chyb při 3D tisku je klíčovým aspektem pro zajištění kvality výsledných výtisků a minimalizaci plýtvání materiálem i časem. V posledních letech se v této oblasti výrazně prosadily metody využívající počítačové vidění a strojové učení, které umožňují automatizované sledování tisku a včasné rozpoznání potenciálních problémů. V této kapitole jsou představeny vybrané odborné práce i dostupná komerční řešení, která se této problematice věnují.

Odborné práce

Využití počítačového vidění pro detekci chyb při 3D tisku je podrobně rozpracováno v práci publikované v časopise Nature Communications. V této práci je navržen systém založený na běžných webkamerách a hluboké neuronové síti s více výstupními hlavami, který umožňuje detekci i automatickou korekci chyb v reálném čase. Systém s názvem CAXTON je trénován na rozsáhlé datové sadě více než 1,2 milionu snímků, které jsou automaticky anotovány podle odchylek od optimálních tiskových parametrů. Díky tomuto přístupu je možné detekovat širokou škálu vad, jako je nesprávný průtok materiálu, chybná výška trysky nebo odchylky teploty, a současně tyto chyby bez zásahu člověka korigovat. Systém byl navržen tak, aby byl použitelný na různých typech tiskáren, s různými materiály a v odlišných geometriích. Studie navíc ukazuje, že použití vícehlavé neuronové sítě umožňuje nejen efektivní detekci, ale i pochopení vzájemných vztahů mezi jednotlivými tiskovými parametry, čímž se zvyšuje robustnost celého řešení [7].

Využití velkých jazykových modelů pro detekci a korekci chyb při 3D tisku bylo podrobně rozpracováno v práci publikované v roce 2024. V této studii je navržen autonomní systém, který využívá běžné webkamery a velký jazykový model (LLM) k analýze snímků tištěného objektu po každé vrstvě. Systém detekuje běžné vady, jako je podextrudování, stringování či deformace vrstev, a na základě těchto zjištění automaticky upravuje tiskové parametry, jako jsou rychlost tisku, průtok materiálu, teplota trysky či výška trysky. Díky hierarchické architektuře multiagentního systému, kde jednotliví agenti spolu komunikují pod dohledem hlavního supervizora, je možné provádět optimalizaci tisku v reálném čase bez nutnosti lidského zásahu [13].

V další práci autoři navrhli systém využívající modely YOLO pro automatickou detekci běžných defektů během FDM tisku, jako je podextrudování, přeextrudování, stringování a tvorba špaget. Systém pracuje v reálném čase a zaměřuje se na dosažení vysoké přesnosti detekce při současném zachování nízké výpočetní náročnosti. Studie se však zaměřila pouze na integraci detekčního systému; v případě odhalení chyby nebylo implementováno zasílání upozornění ani automatické pozastavení tisku [20].

Jiným příkladem použití počítačového vidění pro detekci vad během 3D tisku je práce v níž se autoři zaměřili na detekci vad u víceosého FDM tisku pomocí počítačového vidění a konvolučních neuronových sítí (CNN). Jejich přístup využívá kameru k snímání horní vrstvy tištěného objektu v reálném čase, přičemž tyto obrazy jsou následně klasifikovány pomocí neuronové sítě FDMNet, speciálně navržené pro tuto úlohu. Detekovaným typem vady bylo oddělení vrstev (interlayer stripping), které je v praxi běžné a může výrazně ovlivnit výslednou pevnost a kvalitu výtisku. V rámci experimentu byla vytvořena datová sada složená z reálných obrazů normálních a vadných vrstev, která byla následně rozšířena pomocí datové augmentace. Výsledná přesnost klasifikace dosáhla 83,1%, přičemž citlivost a specifita detekce překročily hodnotu 80 %. Autoři zároveň srovnávali výkon jejich modelu s klasickými sítěmi jako jsou AlexNet a VGG16 a ukázali, že jejich architektura má nižší výpočetní náročnost, což ji činí vhodnou pro nasazení v průmyslovém prostředí [23].

Komerční řešení

Aktuálně jedním z nejznámějších a nejpoužívanějších řešení je Gadget, který je součástí zásuvného modulu OctoEverywhere¹. Gadget byl poprvé zpřístupněn veřejnosti v roce 2022 a jeho základní funkce jsou dostupné zdarma pro všechny uživatele. V případě detekované

¹<https://octoeverywhere.com>

chyby může Gadget pozastavit tisk a upozornit uživatele prostřednictvím různých komunikačních platform, jako je email, Discord nebo Telegram. Nevýhodou tohoto řešení je nižší úroveň soukromí, protože videozáznam z 3D tisku je odeslán na externí servery, kde je analyzován a potenciálně využíván pro trénování modelů. Upozornění navíc neobsahují detailní informace o typu detekované chyby, ani jistotu její správnosti.

Obico je platforma s otevřeným kódem využívající Spaghetti Detective² pro detekci chyb, funguje také jako zásuvný modul do OctoPrintu s využitím vzdáleného serveru pro správu tiskárny a detekci chyb. Zde je možné využít Obico Cloud, který nabízí detekci chyb zdarma pro jednu tiskárnu po dobu 10 hodin měsíčně. Alternativně je možné Obico server hostovat na vlastním zařízení, což poskytuje vyšší úroveň soukromí, ale zároveň vyžaduje technickou znalost a hardware s dostatečným výpočetním výkonem.

Dalším příkladem je Octoprint-detector2³, zásuvný modul vytvořený v rámci bakalářské práce na podobné téma [12]. Tento systém využívá výkon počítače uživatele pomocí knihovny TensorFlow.js, čímž eliminuje potřebu odesílání snímků na vzdálené servery. Nevýhodou je však nutnost mít neustále otevřené okno OctoPrintu v prohlížeči.

Cloud 3D Print⁴ je komerčně dostupný systém, poskytující alternativu k předchozím řešením využívající OctoPrint. Tento systém poskytuje svoje vlastní rozhraní pro připojení a ovládání tiskáren, připojení kamery a detekce chyb pomocí umělé inteligence. Na rozdíl od ostatních, ale nenabízí žádnou možnost detekce chyb při tisku v jejich bezplatném plánu.

Možnosti integrace systémů detekce chyb s 3D tiskárnami

Komerční řešení pro detekci chyb často fungují jako zásuvné moduly pro OctoPrint. Tento přístup nabízí snadnou integraci jak pro vývojáře, tak pro koncové uživatele díky jednotnému rozhraní a široké podpoře tiskáren. Přesto existuje řada zařízení, která s OctoPrintem kompatibilní nejsou – například z důvodu chybějícího USB portu, nebo proto, že jejich vestavěný firmware neumožňuje ovládání přes externí rozhraní. Pro některé uživatele může být právě závislost na OctoPrintu nevýhodou, zejména v případech, kdy výrobce – jako například Bambu Lab – integruje vlastní pokročilé uživatelské rozhraní přímo do systému tiskárny. Vybrané modely Bambu Lab navíc obsahují i vestavěnou kameru s funkcí detekce tvorby špaget. Existují také zcela nezávislá řešení, jako je Cloud 3D Print, která nevyžadují OctoPrint a podporují širokou škálu tiskáren různých výrobců.

Řešení popsaná v odborné literatuře využívají jak přístup založený na integraci do OctoPrintu – například v případě systému CAXTON – tak i přímou nízkourovňovou komunikaci s tiskárnou, jako je tomu u práce využívající velké jazykové modely. V posledním případě, je však systém omezen pouze na tiskárny používající firmware Klipper. Práce využívající modelů YOLO se naopak zaměřila pouze na detekci chyb s využitím nezávisle připojené kamery, bez komunikace s tiskárnou.

Nízkourovňové způsoby komunikace s tiskárnou, například přes sériové rozhraní a standardní G-code příkazy, mají výhodu v tom, že nevyžadují instalaci OctoPrintu ani jiných dalších systémů. To je obzvláště výhodné v případech, kdy použití OctoPrintu není možné nebo žádoucí. Přesto bylo v rámci této práce zvoleno řešení postavené na OctoPrintu, a to především díky jeho podpoře širokého spektra tiskáren, možnosti snadné integrace webkamery a zjednodušenému procesu instalace a správy systému pro běžné uživatele.

²<https://www.obico.io/the-spaghetti-detective.html>

³<https://github.com/mikulash/Octoprint-detector2>

⁴<https://cloud3dprint.com/>

Kapitola 4

Návrh řešení

Tato kapitola se věnuje návrhu a architektuře systému pro detekci chyb při 3D tisku pomocí počítačového vidění. Na základě požadavků stanovených v úvodu práce je cílem navrhnout systém, který bude schopen v reálném čase monitorovat průběh tisku a v případě zjištění chyby vhodným způsobem reagovat.

V první části kapitoly je popsán hlavní cíl systému a způsob, jakým bude dosažen. Následuje podrobný popis architektury systému, který je rozdělen do několika modulů – od detekční logiky, přes upozornění, až po uživatelské rozhraní a volitelnou webovou aplikaci.

Další část se zaměřuje na přípravu datové sady, včetně strategie získávání dat, jejího rozšíření a nástrojů pro anotaci. Poslední oddíl popisuje samotné trénování detekčního modelu a způsoby hodnocení jeho výkonnosti pomocí běžně používaných evaluačních metrik.

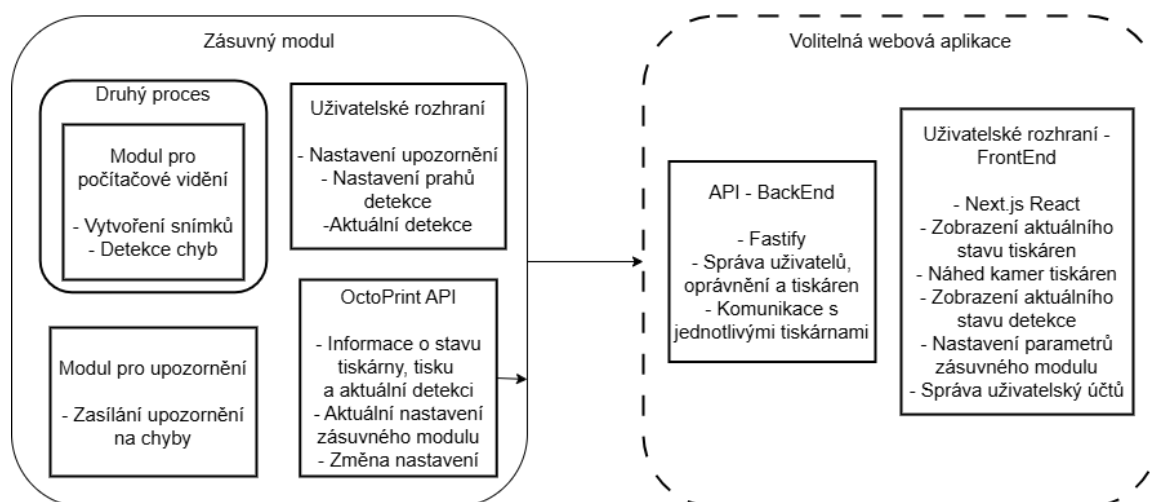
4.1 Popis navrhovaného systému

Cílem navrhovaného systému je automatizované monitorování 3D tisku s využitím počítačového vidění. Systém bude v reálném čase analyzovat snímky z kamery zaměřené na tiskovou plochu a detekovat přítomnost nejčastějších chyb, jako je například stringování nebo špagety. V případě detekce chyby v závislosti na uživatelském nastavení může uživatelé informovat a tisk pozastavit, čímž snižuje riziko znehodnocení výtisku a plýtvání materiálem.

Hlavním rozhraním systému je zásuvný modul (plugin) do aplikace OctoPrint. Tento zásuvný modul je navržen jako jednoduché rozšíření pro jednotlivé tiskárny. Součástí zásuvného modulu je kromě modulu pro detekci chyb také uživatelské rozhraní pro konfiguraci a sledování aktuálního vyhodnocení tisku, a dále modul pro odesílání notifikací. Veškeré zpracování probíhá přímo na zařízení (například Raspberry Pi 5), ke kterému je tiskárna připojena, a bez nutnosti odesílání snímků na externí servery.

K systému je možné volitelně připojit i samostatnou webovou aplikaci, která není součástí zásuvného modulu, ale využívá jeho API. Tato aplikace umožňuje sledování a správu více tiskáren z jednoho místa. Webová aplikace poskytuje přehled o jejich aktuálním stavu, probíhající operaci, náhled připojené kamery nebo konfiguraci zásuvného modulu.

Systém je navržen tak, aby byl schopen chyby detekovat bez připojení k internetu a bez nutnosti připojení dalšího zařízení, jako je například počítač. Důraz je také kladen na soukromí, snímky pro analýzu neopouštějí zařízení uživatele, s výjimkou zaslání upozornění dle uživatelského nastavení. Instalace, konfigurace i samotné používání systému je intuitivní a zvládnutelné během několika jednoduchých kroků.



Obrázek 4.1: Obrázek popisující architekturu

4.2 Architektura systému

Jako základ systému je navržen zásuvný modul pro aplikaci OctoPrint, který bude obsahovat část pro práci se snímky, včetně využití počítačového vidění a modul pro zasílání upozornění. Volitelnou součástí bude také externí webová aplikace umožňující správu více tiskáren z jednoho místa.

Zásuvný modul pro OctoPrint

Zásuvný modul bude představovat hlavní komponentu systému, která bude zajišťovat propojení s tiskárnou a uživatelskou interakci prostřednictvím rozhraní OctoPrintu. Součástí zásuvného modulu bude modul pro počítačové vidění, který bude navržen tak, aby běžel v samostatném procesu odděleném od hlavního vlákna OctoPrintu. Tento přístup zajistí, že výpočetně náročný úkon analýzy obrázku pomocí počítačového vidění nebude ovlivňovat stabilitu a činnost samotného prostředí OctoPrint, jako je například ovládání tiskárny.

Modul pro počítačové vidění

Modul pro počítačové vidění bude zodpovědný za získávání snímků z připojené kamery, jejich předzpracování (například změnu velikosti nebo normalizaci) a následnou analýzu pomocí konvoluční neuronové sítě. Výstupem budou detekované chyby, včetně typu a míry pravděpodobnosti.

Výsledky z modulu pro počítačové vidění budou předávány zpět do hlavního procesu pomocí fronty. V hlavním procesu pak poběží samostatné vlákno, které bude tyto výsledky z fronty průběžně číst a vyhodnocovat. Vlákno provede příslušné akce podle konfigurace uživatele, která zahrnuje nastavení minimální míry jistoty (práhu) pro upozornění nebo pozastavení tisku a volbu, zda má být tisk při detekci chyby automaticky pozastaven.

Modul bude provádět analýzu obrazu z kamery každých pět vteřin a pro provedení akce jako je upozornění nebo pozastavení tisku, bude vyžadovat alespoň dvě po sobě jdoucí detekce. Dále modul bude moci upozornění zasílat s minimálním rozestupem 5 minut a znovu pozastavovat tiskárnu nejdříve po 10 minutách.

Toto sice může způsobit o něco pomalejší reakci na detekovanou chybu, ale umožní to předcházet tomu, aby modul reagoval na falešné detekce, které se objeví pouze v jednom snímku.

Modul pro zasílání upozornění

Součástí zásuvného modulu bude také modul, který bude obstarávat zasílání notifikací na základě požadavků z vlákna zpracovávajícího frontu. Modul bude podporovat více cílů a bude navržen tak, aby byl jednoduše rozšiřitelný o další služby. Nastavení jednotlivých cílů bude možné provést v uživatelském rozhraní OctoPrintu. Jednotlivá upozornění budou obsahovat informaci o tom, zda se jedná pouze o upozornění, nebo zda došlo k pozastavení tisku, jaká chyba byla detekována a s jakou mírou pravděpodobnosti. Součástí zprávy bude také snímek, na kterém byla detekovaná chyba. To umožní uživateli zhodnotit aktuální situaci, bez nutnosti tiskárnu fyzicky navštívit pro kontrolu stavu tisku.

Uživatelské rozhraní zásuvného modulu

Zásuvný modul bude rozšiřovat uživatelské rozhraní aplikace OctoPrint pomocí knihovny Knockout.js, která je standardní součástí jeho systému pro uživatelské rozhraní. Konkrétně bude přidávat nový panel zobrazovaný na hlavní stránce během tisku. Tento panel bude zobrazovat aktuální míru pravděpodobnosti detekované chyby, kterou poskytuje modul pro počítačové vidění. Uživatel tak bude mít možnost v reálném čase sledovat výstup detekčního systému přímo v prostředí OctoPrintu.

Součástí panelu bude také nastavení chování zásuvného modulu. Uživateli dá možnost:

- aktivovat nebo deaktivovat celý zásuvný modul,
- povolit nebo zakázat automatické pozastavení tisku při detekci chyby,
- nastavit minimální hodnotu pravděpodobnosti (práh) pro zasílání upozornění a pro automatické pozastavení tisku.

Tato nastavení se budou aplikovat okamžitě a budou uchovávána v konfiguračních souborech OctoPrintu. Díky přehlednému zobrazení a možnosti rychlé změny konfigurace přímo z hlavní obrazovky se zvýší uživatelský komfort a efektivita reakce na případné chyby během tisku, jelikož všechny informace může mít na jedné stránce.

Stránka s rozšířeným nastavením Zásuvný modul dále přidá do rozhraní OctoPrintu vlastní stránku s rozšířeným nastavením. Na této stránce bude moci uživatel zvolit, která kamera bude pro detekci použita. Možnosti zahrnují buď klasickou kameru nastavenou v OctoPrintu, nebo vlastní kameru. V případě volby vlastní kamery bude muset uživatel zadat adresy pro MJPEG stream a snímky (snapshoty), jak je běžné u externích kamer v prostředí OctoPrintu.

Dále tato stránka poskytne přístup ke stejným nastavením jako panel na hlavní stránce, tedy aktivace modulu, povolení automatického pozastavení a nastavení prahových hodnot. Kromě toho umožní také konfiguraci notifikačních služeb.

Modul bude podporovat dvě notifikační platformy:

- **Discord** – upozornění budou odesílány pomocí Webhook URL, kterou uživatel zadá,

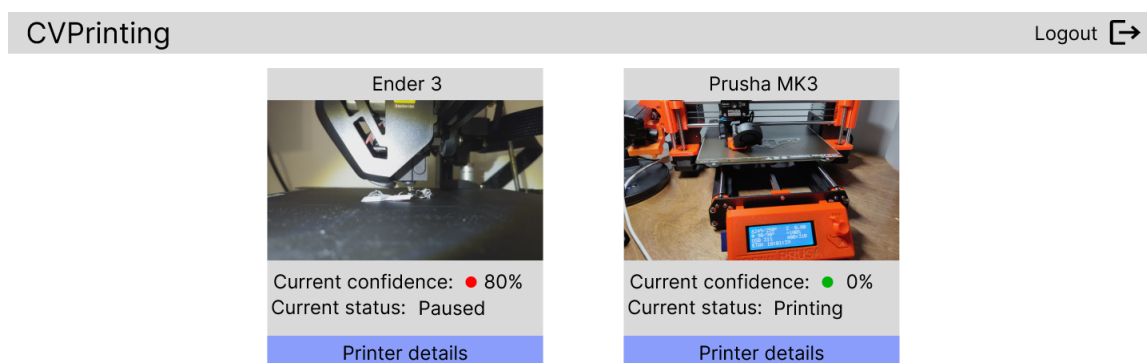
- **Telegram** – komunikace bude probíhat přes účet pro automatizované zasílání zpráv (anglicky *chat bot*), který si uživatel vytvoří dle přiloženého návodu. Uživatel poté může zadat API token a připojit se k tomuto účtu, kterému odešle zprávu, a tím zásuvný modul získá ID chatu, které je nutné pro identifikaci toho, kam se mají upozornění zasílat.

Tímto způsobem bude možné systém přizpůsobit konkrétním potřebám a preferencím uživatele a zajistit doručení upozornění na platformu preferovanou uživatelem.

Volitelná webová aplikace

Volitelným rozšířením systému bude samostatná webová aplikace. Aplikace bude využívat Fastify API pro autentizaci a autorizaci uživatelů a také pro ukládání nastavení jednotlivých tiskáren, které bude potřeba k jejich připojení. Fastify API bude rovněž obstarávat komunikaci s jednotlivými tiskárnami prostřednictvím API poskytnutého aplikací OctoPrint.

Uživatelské rozhraní této aplikace bude realizováno pomocí Next.js frameworku pro React. Tato volba umožní tvorbu responzivního rozhraní pro webovou aplikaci. Uživatelé budou moci pomocí tohoto rozhraní sledovat aktuální stav tiskáren, včetně aktuální míry pravděpodobnosti detekce chyby ze zásuvného modulu. Stejně jako v uživatelském rozhraní v OctoPrintu budou moci uživatelé nastavit prahy pro upozornění a pozastavení tisku, povolení pozastavení tisku, nebo detekční modul zcela vypnout.



Obrázek 4.2: Návrh uživatelského rozhraní volitelné webové aplikace vytvořený pomocí nástroje Figma.

4.3 Příprava datové sady

Pro natrénování modelu pro detekci chyb při 3D tisku je nezbytné vytvořit kvalitní a reprezentativní datovou sadu se snímky různých typů chyb. Vzhledem k tomu, že se jedná o poměrně specifickou oblast, je dostupnost veřejných datových sad omezená a často vykazuje značnou variabilitu v kvalitě, formátu i způsobu anotace.

Získávání obrazových dat

Vlastní datová sada vznikne kombinací několika přístupů. Prvním z nich je manuální sběr snímků z online komunit zaměřených na 3D tisk, jako například fórum Reddit (subfóra jako `r/3Dprinting` a `r/FixMyPrint`). Tyto zdroje často obsahují reálné příklady chyb v různých prostředích, což zajišťuje variabilitu a realističnost trénovacích dat.

Stažené snímky budou filtrovány – odstraní se nerelevantní nebo nekvalitní snímky. Vybrané snímky budou následně anotovány.

Anotace pomocí Label Studio a Segment Anything Model

K anotaci bude použit nástroj Label Studio, který umožňuje efektivní ruční označování objektů na snímcích. Pro urychlení tohoto procesu bude využit také Segment Anything Model (SAM) od společnosti Meta, který umožňuje poloautomatickou segmentaci objektů na základě minimálního uživatelského vstupu. Tím se výrazně zkracuje čas potřebný k anotaci složitějších snímků.

Výstupem budou segmentační masky s označeným typem chyby (například stringování, špagety, přeextrudování), které bude možné přímo použít při tréninku detekčního modelu.

Využití veřejně dostupných datových sad

Dalším zdrojem jsou existující veřejné datové sady, dostupných například na platformách jako Kaggle nebo Roboflow. Tyto datové sady často obsahují stovky až tisíce anotovaných snímků a mohou pokrývat běžné typy chyb. Jejich využití však přináší několik potenciálních problémů:

- **Nekonzistentní anotace:** Různé datové sady používají různé formáty anotací (například COCO, YOLO, Pascal VOC), které je nutné převádět a sjednocovat.
- **Nejasná kvalita:** Některé datové sady mohou obsahovat chybně anotované, nebo nekvalitní snímky.
- **Rozdílné rozdělení:** Dalším problémem je, že existující datové sady mohou mít jinak rozdělené snímky, například pouze na výtisky s chybou a výtisky bez chyby. Taková datová sada by pak vyžadovala ručně roztrždit všechny snímky podle typu zobrazené chyby a upravit jejich anotace.
- **Datová sada pro jinou úlohu:** Ne všechny datové sady obsahují anotace vytvořené pro segmentaci, velmi časté jsou datové sady pro klasifikaci nebo detekci objektů.

Z výše uvedených důvodů budou tyto datové sady využity pouze jako doplněk k vlastní datové sadě, a to po pečlivém zhodnocení jejich kvality a relevance.

Cíle a kvalita datové sady

Cílem je vytvořit datovou sadu s dostatečnou rozmanitostí scén, světelných podmínek, druhů chyb a úhlů pohledu, aby model dokázal generalizovat na reálné situace. Datová sada může být průběžně rozšiřována, což umožní postupně trénovat přesnější modely pro detekci chyb při 3D tisku.

Součástí datové sady budou anotace jak pro výtisky bez chyb, tak i pro nejčastější typy chybm, včetně stringování, špaget, přeextrudování, podextrudování a mezer ve výtisku

(například praskání nebo díry mezi vrstvami). Tato kategorizace umožní modelu lépe rozlišit různé typy chyb a upozornit tak uživatele na výskyt konkrétní chyby.

Rozšíření dat

Aby se zvýšila robustnost modelu a jeho schopnost správně fungovat na nové reálné situace, bude na trénovací sadu aplikována tzv. augmentace dat. Pomocí různých transformačních operací budou ze stávajících snímků vytvářeny nové varianty, čímž se efektivně rozšíří velikost a variabilita trénovací množiny.

Dostupné techniky rozšíření dat zahrnují například:

- změnu jasu a kontrastu,
- rotaci a horizontální/vertikální převrácení,
- náhodný výřez (cropping), který simuluje situace, kdy není vidět celý výtisk – to pomáhá zvýšit přesnost detekce,
- přidání šumu nebo rozostření – technika důležitá pro zlepšení přesnosti při následném použití webkamer, které mají obecně nižší kvalitu,
- změnu barevného prostoru (například na černobílý).

Tyto techniky jsou prováděny během trénování modelu v reálném čase pomocí knihovny Albumentations, která je využívána v rámci frameworku Ultralytics, jenž umožňuje trénování YOLO modelů pro počítačové vidění. [9]

Stav vytvořené datové sady

Datová sada byla průběžně rozšiřována v celém průběhu práce na bakalářském projektu. Její základ tvoří ručně získané snímky z různorodých zdrojů – internetu, 3D tiskáren umístěných v Open Space na Fakultě informačních technologií Vysokého učení technického v Brně a vlastních tisků. Tyto snímky zachycují jak bezchybné výtisky, tak různé typy chyb typické pro FDM tisk, konkrétně: špagety, stringování, přeextrudování, podextrudování a mezery ve výtisku.

Datová sada byla dále rozšířena o snímky pocházející z veřejně dostupné kolekce umístěné na platformě Kaggle¹, která byla vytvořena v rámci jiné bakalářské práce. Tyto snímky sloužily jako doplňkový zdroj pro reprezentaci jednotlivých chyb.

Další část snímků pochází z platformy Roboflow², přičemž z této sady byly využity pouze snímky bezchybného tisku, případně snímky obsahující chyby typu špagety a stringování. Výhodou této sady je, že již obsahuje předpřipravené segmentační masky, které mohly být převzaty bez nutnosti další ruční anotace.

Naopak snímky z vlastních zdrojů a z Kaggle datové sady bylo nutné manuálně anotovat vytvořením segmentačních masek.

Celková datová sada obsahuje následující počty anotací pro jednotlivé třídy:

- **PrintOk** – 182 anotací bezchybného tisku
- **Stringing** – 460 anotací se stringováním

¹<https://www.kaggle.com/datasets/mikulhe/3d-printing-errors>

²<https://universe.roboflow.com/estuy1/anomalydetection-fynn9>

- **Spaghetti** – 351 anotací se špagetami
- **OverExtrusion** – 130 anotací s přeextrudováním
- **UnderExtrusion** – 94 anotací s podextrudováním
- **Gaps** – 111 anotací s mezerami ve výtisku

Datová sada rovněž obsahuje 398 tzv. *background* snímků, které nebyly anotovány žádnou chybovou třídou. Tyto snímky často obsahují výtisky bez viditelné chyby a byly převzaty zejména z datové sady pocházející z platformy RoboFlow. Slouží jako negativní vzorky při trénování modelu a zvyšují jeho robustnost vůči falešně pozitivním detekcím.

Pro účely trénování modelu byla datová sada rozdělena přibližně v poměru 80 % pro trénování, 10 % pro validaci a 10 % pro testování. Toto rozdělení bylo provedeno náhodně, přičemž bylo zajištěno rovnoměrné zastoupení všech tříd v jednotlivých podmnožinách.

Datová sada tak představuje různorodý a realistický základ pro trénink modelu detekujícího chyby během 3D tisku pomocí metod počítačového vidění.

4.4 Trénování modelu

Po dokončení přípravy a anotace datové sady následuje fáze trénování modelu. Cílem je vytvořit konvoluční neuronovou síť schopnou detekovat chyby při 3D tisku na základě snímků pořízených kamerou. Proces trénování probíhá ve více krocích: načtení dat, načtení předtrénovaného modelu nebo architektury sítě, specifikace trénovacích parametrů a spuštění trénování.

Model je trénován v prostředí Kaggle, které poskytuje cloudové GPU akcelerované výpočetní prostředí dostupné prostřednictvím interaktivních Python notebooků. Tato platforma umožňuje rychle iterovat nad návrhem architektury, laděním parametrů i testováním výsledků bez potřeby vlastního výkonného hardwaru. Výhodou je také možnost jednoduše sdílet notebooky a porovnávat experimenty.

Před trénováním jsou obrazová data normalizována a případně augmentována. Součástí standardizace je přizpůsobení velikosti obrázků na jednotný rozměr odpovídající požadavkům použité konvoluční sítě a normalizaci hodnot jednotlivých pixelů do rozsahu podle použité architektury. Následuje návrh detekční architektury – buď vlastní sítě, nebo využití předtrénovaného modelu, který je následně doladěn (anglicky *fine-tuned*) na specifickou doménu chyb 3D tisku.

Trénování probíhá v několika epochách, během nichž se model postupně učí minimalizovat hodnotu ztrátové funkce (anglicky *Loss function*), která vyjadřuje míru chyby predikce, a to za pomoci optimalizačního algoritmu, jako je například Adam nebo SGD. Klíčové hyperparametry jako počet epoch, velikost vstupních obrázků nebo nastavení optimalizátoru zásadně ovlivňují výkonnost výsledného modelu. Vybrané hyperparametry jsou detailněji popsány v následující části. Hlavním zdrojem pro tuto kapitolu je [9]

Nastavení hyperparametrů a konfigurace pro trénování

Hlavním zdrojem pro tuto část je [9].

- **Počet epoch** – Jedna epocha se rovná jedné trénovací iteraci, při které model vidí všechny snímky z trénovací sady. Vyšší počet epoch umožňuje modelu lépe se naučit vlastnosti z dat. Příliš mnoho epoch však může způsobit přetrénování modelu, což

je situace, kdy se model velmi dobře naučí rozpoznávat objekty v trénovací sadě, ale bude mít horší výsledky na validační a testovací sadě.

- **Velikost obrázku** – Konvoluční neuronová síť na svém vstupu vyžaduje obrázky o stejné velikosti jako její první vrstva. To znamená, že pokud má první vrstva velikost 640×640 , je nutné, aby všechny snímky na jejím vstupu měly právě tuto velikost. Snímky lze do požadovaného rozměru přizpůsobit několika způsoby – například jejich roztáhnutím, přičemž ale není zachován poměr stran, což vede k deformaci. Častější možností je, že se delší strana obrázku přizpůsobí cílové velikosti a zbývající prostor podél kratší strany se doplní černými pixely. Tím zůstává zachován poměr stran a nedochází ke zkreslení obrázku.
- **Rychlost učení (anglicky *Learning rate*)** – Důležitým hyperparametrem, který má přímý vliv na úspěšnost trénování neuronové sítě, je rychlost učení. Tento parametr určuje velikost kroků, jimiž se během trénování aktualizují váhy neuronů s cílem minimalizovat ztrátovou funkci. Příliš vysoká hodnota může sice zrychlit proces učení, avšak často vede k oscilacím chybové funkce, což může způsobit, že model nedosáhne optimální chyby. Naopak příliš nízká hodnota sice umožňuje stabilní konvergenci, ale výrazně zpomaluje trénování. Mezi běžně používané počáteční hodnoty patří například 0,1 nebo 0,01. V praxi se často uplatňuje také postupné snižování rychlosti učení (anglicky *learning rate decay*) v průběhu trénování, případně adaptivní přístupy implementované v optimalizačních algoritmech, které hodnotu learning rate automaticky přizpůsobují na základě charakteru ztrátové funkce. Mezi běžně používané optimalizátory, které tento parametr upravují dynamicky, patří Adam, RMSProp nebo novější AdamW.
- **Velikost dávky (anglicky *Batch size*)** – Volba velikosti dávky představuje důležité rozhodnutí při trénování neuronových sítí, které ovlivňuje jak kvalitu, tak rychlost učení. Použití celé trénovací množiny na výpočet gradientu, tzv. batch gradient descent, poskytuje přesný výpočet, ale je výpočetně náročné. Naopak trénování na základě jednotlivých vzorků (stochastic nebo online gradient descent) je výpočetně efektivní, ale často vede k větší varianci v odhadu gradientu. Kompromisem je tzv. malá dávka (anglicky *minibatch* nebo *minibatch stochastic gradient descent*), který využívá menší podmnožiny dat pro jeden krok učení. Velikost těchto malých dávek se obvykle volí s ohledem na několik faktorů – větší dávky poskytují přesnější odhad gradientu, ale přinášejí pouze omezené zrychlení; malé dávky mohou mít regulační efekt díky přidanému šumu, který zlepšuje zobecnění modelu, ale trénování může trvat delší dobu z důvodu potřeby většího počtu trénovacích kroků pro projití celé datové sady. Volba velikosti dávky je rovněž ovlivněna výpočetní architekturou – příliš malé dávky nevyužívají efektivně vícejádrové systémy, zatímco příliš velké dávky mohou překročit kapacitní limity paměti. U GPU je také obvyklé používat velikosti dávky, které jsou mocninami dvou (například 32, 64, 128), které často vedou k efektivnějšímu výpočtu. [11]
- **Vrstvy sítě k zmrazení** – Při využití předtrénovaných modelů je klíčové rozhodnutí, zda a kolik vrstev modelu „zmrazit“ (tj. ponechat jejich váhy nezměnné). Při tomto rozhodování je vhodné řídit se především velikostí a podobností cílové datové sady vůči datům, na nichž byl model původně natrénován. Pokud je cílová doména podobná původní, doporučuje se zmrazit větší množství vrstev modelu, zejména vrstvy,

kteře extrahují obecné rysy (například hrany či textury), jelikož tyto vrstvy obsahují dobře přenositelné reprezentace. Naopak v případě výrazně odlišných domén nebo při dostatečně velkém objemu dat je vhodné ponechat více vrstev trénovatelných. Tento hyperparametr je vhodné nastavovat experimentálně na základě testování s ohledem na konkrétní úlohu a dostupná data.

Výsledný model je pravidelně validován pomocí datové sady a na základě dosažených výsledků lze dále upravovat trénovací parametry a nebo datovou sadu.

Použití prostředí Kaggle umožňuje celý proces urychlit a opakovat, což je výhodné pro ladění trénovacího procesu.

Výsledky trénování modelu

Pro účely této práce byly zvoleny modely YOLO od Ultralytics, a to především díky jejich široké nabídce architektur určených pro různé úlohy počítačového vidění – od klasifikace přes detekci objektů až po segmentaci. Knihovna *Ultralytics* zároveň poskytuje řadu nástrojů pro trénování, validaci a export modelů do různých formátů. Umožňuje také porovnávat výkonost modelů v různých prostředích díky podpoře formátů jako TensorFlow.js, ONNX nebo MNN, což napomáhá zvolit optimální formát pro konkrétní cílové zařízení.

V rámci testování byly porovnávány dvě nejnovější verze modelů – YOLOv11 a YOLOv12. Model YOLOv11 byl vydán 30. září 2024, zatímco YOLOv12 následoval 20. února 2025. Obě verze nabízejí několik variant dle velikosti modelu (označení *n*, *s*, *m*, *l*, *x*), přičemž menší varianty jsou vhodné zejména pro vestavěná zařízení s omezeným výpočetním výkonem. YOLOv11 poskytuje předtrénované modely jak pro detekci objektů, tak i pro segmentaci, zatímco YOLOv12 zatím nabízí pouze modely pro detekci. Všechny modely byly jejich autory předtrénovány na standardní datové sadě COCO.

Před samotným trénováním bylo nutné datovou sadu upravit. Ukázalo se, že modely vykazovaly nízkou přesnost u tříd jako přeextrudování, podextrudování a mezery, což bylo způsobeno jejich nízkým zastoupením v datech. Z tohoto důvodu bylo rozhodnuto trénovat modely finálně pouze na detekci vad typu *špagety* a *stringování*. Jelikož jedna z použitých datových sad snímky bez závad neoznačovala jako *Print OK*, ale používala je jako tzv. *background* snímky (tj. snímky bez anotací), bylo proto nutné tyto snímky zbavit anotací a ponechat je jako negativní vzorky.

Na základě několika trénovacích pokusů s různým nastavením hyperparametrů bylo finální nastavení určeno následovně: počet epoch byl stanoven na 100, velikost vstupního obrázku na 640 a parametr trpělivosti (anglicky *patience*), který ukončuje trénování v případě, že nedojde k žádnému zlepšení během definovaného počtu epoch, byl nastaven na hodnotu 30. Parametr *freeze*, určující počet zmrazených počátečních vrstev, byl nastaven na 10. Během experimentů byly testovány různé hodnoty pro velikost dávky, rychlost učení i optimalizační algoritmy. Výsledky těchto trénovacích pokusů byly následně porovnány s výsledky dosaženými při použití hyperparametrů automaticky nastavených knihovnou *Ultralytics*. Ve všech případech bylo při automatickém nastavení dosaženo lepších výsledků, a proto byly tyto hodnoty ponechány i při finálním trénování modelů.

Po ukončení trénování – ať už po dosažení 100 epoch, nebo po vyčerpání trpělivosti – knihovna automaticky vybere nejlepší uložený model na základě vlastní metriky označované jako *fitness*. Tato metrika kombinuje dvě hodnoty *mean Average Precision* (mAP) s různou vahou: mAP@0.5 a mAP@0.5:0.95. Výpočet metriky fitness probíhá podle následujícího vzorce:

$$\text{fitness} = \text{mAP}_{@0.5} \cdot 0,1 + \text{mAP}_{@0.5:0.95} \cdot 0,9$$

Tabulka 4.1 shrnuje výsledky jednotlivých modelů z hlediska dosažené validační a testovací přesnosti. Tyto hodnoty byly získány po dokončení trénování v původní implementaci frameworku PyTorch. Doba zpracování jednoho snímku byla měřena na modelech převedených do formátu MNN a spuštěných na zařízení Raspberry Pi 5 s 4 GB RAM.

Model	Validační P	Testovací P	Doba zpracování (MNN)
Yolo11n	60 %	48 %	101 ms
Yolo11s	67 %	58 %	252 ms
Yolo11m	61 %	56 %	701 ms
Yolo11n-seg	63 %	49 %	135 ms
Yolo11s-seg	61 %	49 %	327 ms
Yolo12n	58 %	49 %	151 ms
Yolo12s	47 %	47 %	359 ms

Tabulka 4.1: Porovnání jednotlivých modelů z hlediska přesnosti validace, testování a doby zpracování jednoho snímku.

Dle výsledků na validační a testovací datové sadě dosahuje nejlepších výsledků model pro detekci objektů Yolo11s. Tento model byl proto vybrán pro detekci chyb v zásuvném modulu.

Ještě před integrací modelu počítačového vidění do zásuvného modulu byla využita funkce **benchmark** z knihovny Ultralytics, pomocí níž byl na platformě Raspberry Pi identifikován nejvhodnější framework s nejvyšší výkonností. Výsledky tohoto testu jsou uvedeny v tabulce 4.2.

Framework	Doba zpracování (ms/obrázek)	FPS
TorchScript	1172,88	0,85
PyTorch	1108,43	0,9
PaddlePaddle	1086,38	0,92
ONNX	454,07	2,2
MNN	253,26	3,95
NCNN	185,93	5,38
OpenVINO	185,29	5,4

Tabulka 4.2: Výsledky testu (benchmarku) – srovnání frameworků podle doby potřebné ke zpracování jednoho snímku a maximální dosažitelné snímkové frekvence (FPS). Z tabulky byly vypuštěny frameworky nekompatibilní s platformou Raspberry Pi 5.

Ačkoli závěrečný test ukazuje, že frameworky OpenVINO a NCNN dosahují nejlepších výsledků z hlediska rychlosti, volba frameworku MNN byla provedena již v dřívější fázi vývoje a testování, kdy framework OpenVINO vykazoval problémy při implementaci na cílové platformě (Raspberry Pi 5).

Z tohoto důvodu byl pro další vývoj zvolen framework MNN, který kromě stále dobrých výsledků nabízel i jednoduchou integraci do zásuvného modulu.

Kapitola 5

Implementace

V této kapitole je popsána konkrétní realizace systému navrženého v předchozí části práce. Postupně jsou představeny jednotlivé komponenty, jejich struktura, použitá technologie a způsob jejich vzájemné komunikace.

Nejprve je popsána implementace zásuvného modulu pro OctoPrint, včetně způsobu integrace modulu pro počítačové vidění a uživatelského rozhraní zásuvného modulu. Následně je popsán průběh testování zásuvného modulu na datech získaných z tiskáren v OpenSpace a z platformy YouTube. Tato část zahrnuje také testování na skutečných tiscích bez chyb i se simulovanými chybami, včetně testování s pohledem shora a vyhodnocení dosažených výsledků.

V závěru kapitoly je popsána implementace volitelné webové aplikace, konkrétně její serverové části (backendu) a uživatelského rozhraní (frontendu).

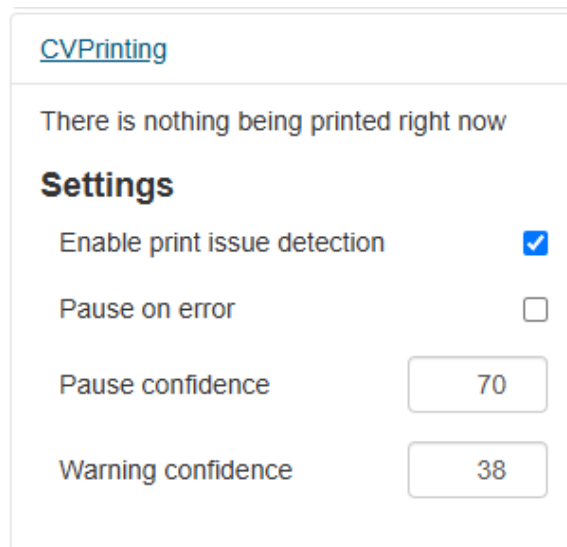
5.1 Zásuvný modul

Vlastní implementace zásuvného modulu pro OctoPrint je obsažena ve složce s názvem `octoprint_cvprinting`, která se skládá z několika souborů a podadresářů, z nichž každý plní specifickou roli v rámci celého systému.

Základní inicializace zásuvného modulu probíhá v souboru `__init__.py`. Tento soubor definuje hlavní třídu zásuvného modulu, která zajišťuje inicializaci nastavení a dědí rozhraní z OctoPrintu pro tvorbu API koncových bodů (anglicky *endpoint*), správu nastavení či reakce na události, jako je spuštění nebo pozastavení tisku.

Samotná analýza snímků probíhá ve zvláštním procesu, který je vytvořen pomocí třídy `multiprocessing.Process`. V tomto procesu je spuštěna instance třídy obsažená v souboru `monitoring.py`. Tento proces periodicky (každých pět sekund) volá funkci pro analýzu snímku ze souboru `visionModule.py`, který obsahuje kód pro počítačové vidění. Zvolený interval představuje kompromis mezi rychlostí reakce na chybu a výpočetní náročností celého procesu. Výsledky analýzy jsou poté prostřednictvím fronty předány zpět hlavnímu procesu.

Odesílání upozornění na externí služby, jako je Discord nebo Telegram, je řešeno v souboru `notifications.py`. Tento modul poskytuje funkce pro formátování a odesílání zpráv na základě výsledků analýzy. Díky modulárnímu návrhu lze systém jednoduše rozšířit o podporu dalších platforem pro zasílání upozornění, což umožňuje přizpůsobení různým preferencím uživatelů.



Obrázek 5.1: Postraní panel v OctoPrintu

Rozdělení implementace do několika samostatných souborů zajišťuje lepší přehlednost, modularitu a snadnější rozšiřitelnost systému.

Složka `data` obsahuje předtrénovaný model počítačového vidění ve formátu pro knihovnu MNN, který je využíván při detekci chyb během tisku.

Uživatelské rozhraní zásuvného modulu je tvořeno pomocí šablon Jinja2, které se nachází ve složce `templates`. Tyto šablony definují HTML kód pro stránku s nastaveními modulu a pro panel zobrazený v postranním panelu OctoPrintu. Interaktivitu těchto stránek zajišťují JavaScriptové soubory umístěné ve složce `static`.

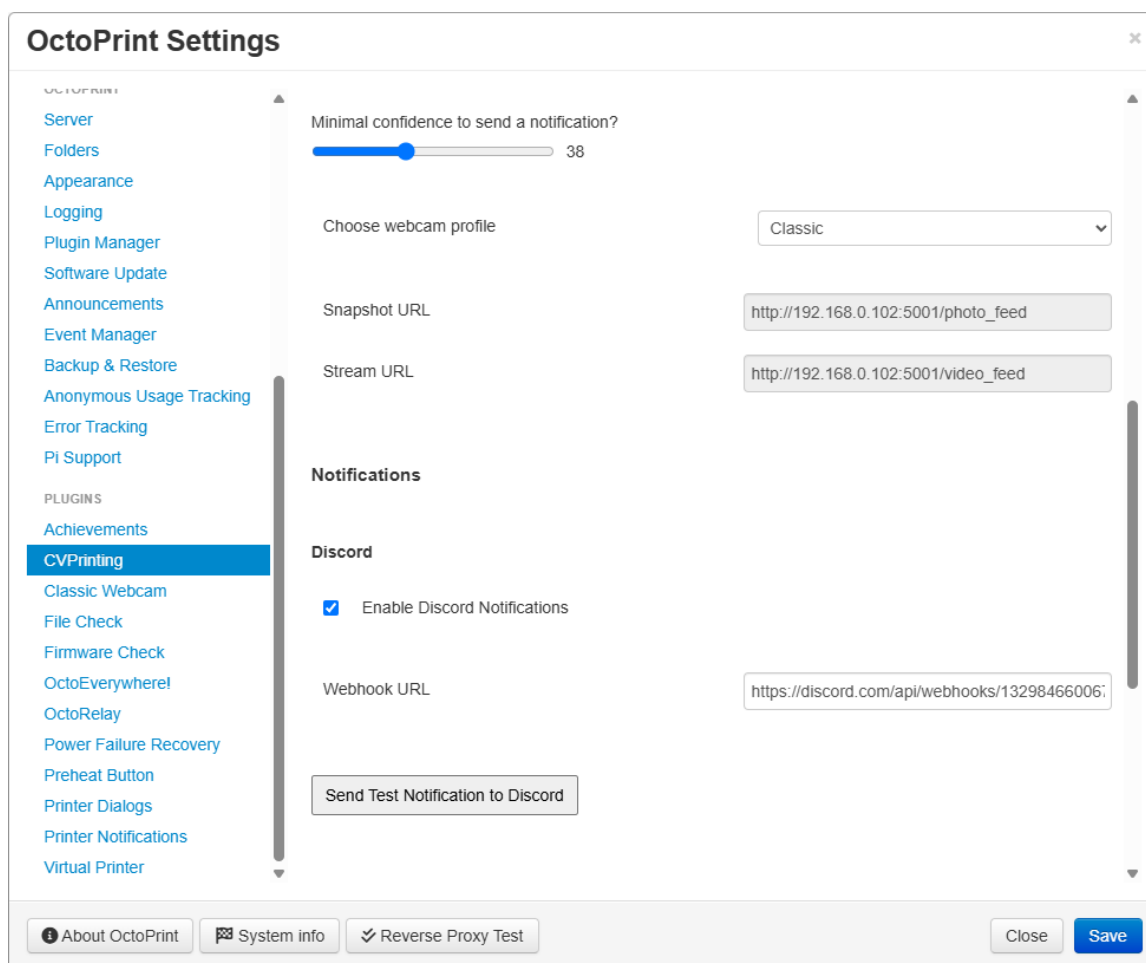
Implementace modulu pro analýzu obrazových dat

Na základě návrhové architektury popsané v předchozí kapitole byla implementována samostatná komponenta zajišťující analýzu obrazových dat pomocí počítačového vidění.

Prvotní implementace modulu pro analýzu obrazových dat byla realizována za využití samostatného vlákna vytvořeného pomocí knihovny `threading`. Tato implementace se však během testování ukázala jako nevhodná, neboť při jejím nasazení docházelo ke znatelnému zpomalování a přerušování chodu tiskárny ve chvílích, kdy modul prováděl výpočetně náročné zpracování obrazových snímků z kamery. To způsobilo sníženou kvalitu výtisků, jelikož při zastavení pohybu tiskárny došlo k natavení plastu v daném místě. Identifikovaný problém pramenil ze způsobu, jakým je v interpretru CPython implementováno vícevláknové zpracování.

Konkrétně se jedná o přítomnost tzv. globálního zámku interpretu (anglicky *Global Interpreter Lock*, zkr. GIL), jehož cílem je zajištění bezpečného přístupu ke sdíleným strukturám v paměti. Tento zámek však zároveň znemožňuje skutečný paralelismus při provádění Python kódu ve více vláknech, protože v daný okamžik může vykonávat instrukce pouze jedno vlákno. V důsledku toho nelze efektivně využít více jader procesoru pro paralelní výpočty, což se negativně odrazilo na plynulosti samotného tisku [2].

Z těchto důvodů byla zvolena alternativní architektura založená na vícenásobných procesech. Analýza obrazových dat byla přesunuta do samostatného procesu, který je inicializován při zahájení tisku. Hlavní proces zároveň inicializuje pomocné vlákno, jehož úkolem



Obrázek 5.2: Část stránky s nastavením pro webkameru a discord upozornění pro zásuvný modul

je čtení zpráv z fronty implementované prostřednictvím knihovny `multiprocessing`. Tato knihovna zároveň umožňuje efektivní sdílení dat mezi procesy – v rámci návrhu je využíván tzv. sdílený manažerský slovník, který uchovává aktuální nastavení webkamer a umožňuje jeho dynamickou aktualizaci i během samotného procesu tisku. Dále je využita sdílená logická proměnná, která slouží k detekci požadavku na ukončení podřízeného procesu, a to jak při pozastavení, dokončení, tak i při předčasném ukončení tisku.

Veškeré nové procesy jsou vytvářeny výhradně prostřednictvím metody `spawn`. Tento způsob inicializace zajišťuje vytvoření čistého a izolovaného procesu, čímž se eliminuje riziko dědění nežádoucího stavu zámek z nadřazeného procesu. V opačném případě, například při použití metody `fork`, může dojít ke kopírování paměťového prostoru, včetně interního stavu zámek, což v kombinaci s vícevláknovým prostředím často vede ke vzniku tzv. deadlocků. Výhodou metody `spawn` je také její multiplatformní podpora – je dostupná na systémech typu Windows, Linux i macOS, čímž je zajištěna širší přenositelnost celé aplikace. [6]

Snímek je nejprve stažen z vybrané webkamery a prostřednictvím URL adresy pro snímky. Po jeho úspěšném načtení je provedeno předzpracování, během něhož je obraz upraven na rozměry 640×640 pixelů. V případě, že má původní snímek jiný poměr stran, je tento rozdíl kompenzován přidáním výplně (paddingu) v podobě černých pixelů, čímž se zachová poměr stran bez deformace obrazu. Následně je obraz převeden z barevného prostoru BGR do formátu RGB, který odpovídá požadavkům použité neuronové sítě.

Samotná inferenční část modulu je realizována s využitím knihovny MNN (Mobile Neural Network) vyvíjené společností Alibaba. Jedná se o vysoce optimalizovaný framework pro inferenci neuronových sítí, který je navržen pro nasazení na zařízeních s omezeným výpočetním výkonem, jako jsou mobilní nebo vestavěná zařízení. Knihovna nabízí široké možnosti pro efektivní práci s předzpracováním dat, včetně vlastního modulu `MNN.cv`, který umožňuje převod obrazu do tenzorové reprezentace, normalizaci vstupních dat a jejich přípravu pro inferenci.

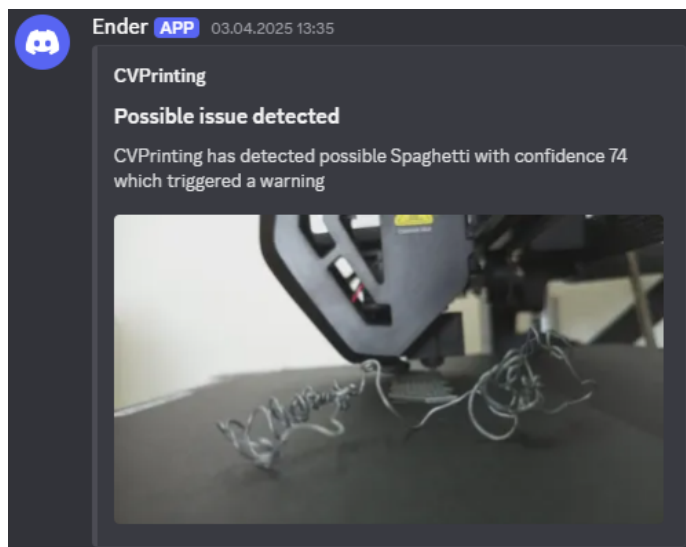
Výsledkem analýzy je výběr třídy s nejvyšší pravděpodobností, která reprezentuje typ detekované chyby na daném snímku. Tato hodnota je následně vložena do sdílené fronty, odkud si ji hlavní proces může přečíst. Na základě nastavených podmínek pak může zaslat upozornění, nebo tisk pozastavit.

Celý zásuvný modul je navržen s důrazem na stabilitu, modularitu a snadné přizpůsobení potřebám uživatele. Díky zvoleným technologiím a architektuře je možné jej efektivně provozovat i na zařízeních s omezeným výpočetním výkonem, jako je například Raspberry Pi 5, a zároveň jej dále rozšiřovat podle specifických požadavků. Tento přístup zajišťuje nejen vyšší výkonnost, ale i celkovou robustnost systému, který tak může běžet stabilně bez negativního dopadu na samotný proces tisku.

5.2 Testování modelu a uveřejnění zásuvného modulu

Zásuvný modul byl testován ve více fázích a za různých podmínek, aby byla ověřena jeho funkčnost, spolehlivost a přínos pro detekci chyb během 3D tisku. Prvotní testování probíhalo pomocí časosběrných videí 3D tisků dostupných na platformě YouTube a dále na záznamech pořízených v rámci OpenSpace na Fakultě informačních technologií VUT. Tato videa posloužila k ověření schopnosti detekčního modelu identifikovat běžné chyby tisku v různých světelných podmínkách a z různých úhlů.

Po úspěšném ověření detekční části následovalo testování celého zásuvného modulu v reálném prostředí. Toto testování je popsáno v následující kapitole 6.



Obrázek 5.3: Upozornění odeslané na platformu Discord



Obrázek 5.4: Upozornění s pauzou odeslané na platformu Telegram

Zásuvný modul byl začátkem dubna zpřístupněn veřejnosti prostřednictvím platformy GitHub¹ a následně sdílen také na oficiálním Discord serveru pro komunitu OctoPrint. Po dalším interním i komunitním testování byl zásuvný modul zveřejněn prostřednictvím integrovaného správce zásuvných modulů přímo v rozhraní OctoPrint². Což umožňuje jeho jednoduchou instalaci přímo z uživatelského prostředí bez nutnosti ručního zásahu.

V rámci počáteční fáze testování byly zpracovány všechny dostupné záznamy 3D tisků pořízené v OpenSpace na Fakultě informačních technologií VUT. Tato videa byla pořízena za různých světelných podmínek a s rozdílnými tisky, což umožnilo ověřit chování detekčního modelu v rozmanitých situacích. Z každého videa byl extrahován jeden snímek za sekundu, čímž vzniklo přibližně 11 tisíc snímků. Tato frekvence extrakce byla zvolena jako kompromis mezi dostatečným pokrytím průběhu tisku a zpracovatelností dat. Každý z těchto snímků byl následně analyzován pomocí vybraného detekčního modelu YOLO, přičemž byly ucho-

¹<https://github.com/Spini11/OctoPrint-Cvprinting>

²<https://plugins.octoprint.org/plugins/cvprinting/>

vány pouze ty, u nichž došlo k pozitivní detekci. Výsledkem tohoto procesu bylo 62 snímků obsahujících alespoň jednu detekci spolu s příslušnými ohraničujícími rámečky a popisky, které byly vizuálně zkontrolovány.

Z celkového počtu 62 snímků se 27 případů ukázalo jako platné detekce závady typu špagety, což představuje přibližně 44 % všech pozitivních detekcí. Zbývající detekce představovaly falešně pozitivní případy, nejčastěji způsobené rušivými objekty v pozadí scény, které vykazovaly podobné vizuální rysy jako sledované závady. Ve většině těchto případů šlo o jiné 3D výtisky nacházející se v blízkosti testované tiskárny nebo o části sousedních tiskáren, které se dostaly do záběru kamery.

Je důležité poznamenat, že při tomto testování nebyla využita žádná z metod pro omezení falešných detekcí, které jsou implementovány přímo v zásuvném modulu – konkrétně požadavek na výskyt detekce alespoň ve dvou po sobě jdoucích snímcích před vyvoláním akce. Tento mechanismus slouží v běžném provozu k eliminaci náhodných chyb a zvýšení celkové spolehlivosti systému. Počet detekcí je tedy v rámci této testovací fáze vyšší než za běžného provozu při použití zásuvného modulu.

5.3 Serverová část pro webovou aplikaci

Serverová část webové aplikace je implementována v souboru `server.js`, který obsahuje jak definici jednotlivých REST API koncových bodů, tak i kód pro inicializaci lokální databáze využívající systém `SQLite`. Serverová část je navržena s důrazem na jednoduchost, přehlednost a snadnou rozšiřitelnost, přičemž plně podporuje víceúrovňový systém oprávnění.

Databáze se inicializuje automaticky při spuštění serveru a obsahuje tabulky pro ukládání informací o uživateli, tiskárnách a jejich přiřazení, konfiguracích zásuvného modulu či historii přístupů. Často používané SQL operace jsou pak přeneseny do samostatných funkcí, umožňujících jejich znovupoužití.

Autentifikace uživatelů probíhá prostřednictvím JWT (JSON Web Token), který je po úspěšném přihlášení uložen do cookie a následně využíván při ověřování oprávnění během práce s API. Validace tokenu a zajištění bezpečného přístupu je implementováno pomocí middleware, který je aplikován na chráněné koncové body.

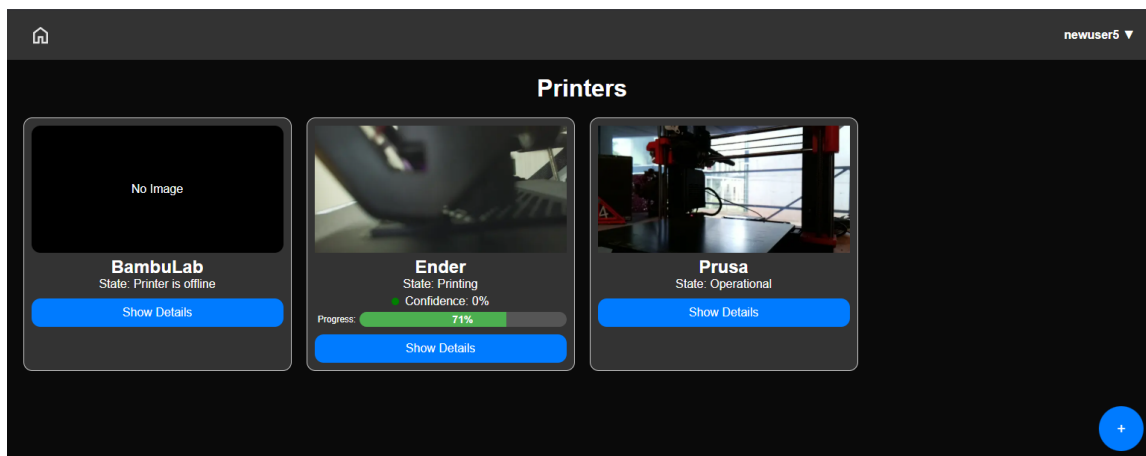
Systém rozlišuje tři úrovně oprávnění uživatelů:

- **Tvůrce** – role automaticky přiřazená prvnímu registrovanému uživateli. Má plnou kontrolu nad aplikací a oprávnění spravovat ostatní účty i tiskárny.
- **Administrátor** – může vytvářet nové uživatelské i administrátorské účty, měnit hesla, upravovat tiskárny a mazat uživatelské profily.
- **Uživatel** – má přístup k tiskárnám, ke kterým mu byla udělena oprávnění, a může měnit konfiguraci modulu počítačového vidění pro tyto tiskárny.

Veškerá hesla jsou před uložením do databáze bezpečně hašována pomocí knihovny `bcrypt`, čímž je zajištěna ochrana citlivých údajů i v případě úniku databáze.

Do systému lze přidávat i tiskárny připojené prostřednictvím OctoPrintu, a to nezávisle na přítomnosti zásuvného modulu CVPrinting. Funkce systému budou bez přítomnosti tohoto zásuvného modulu značně omezené.

Serverová část dále obsahuje funkce pro získávání informací o aktuálním stavu tiskáren prostřednictvím API rozhraní poskytnutého OctoPrintem. Tyto požadavky jsou odesílány



Obrázek 5.5: Zobrazení seznamu tiskáren

periodicky v intervalu pěti sekund, což umožňuje webové aplikaci zobrazovat vždy aktuální stav jednotlivých zařízení.

Samotné testy API koncových bodů se nacházejí ve složce `tests` a pokrývají klíčové scénáře přihlášení, správy účtů, přístupových práv i základní operace s tiskárnami. Některé koncové body nebylo možné automaticky otestovat kvůli jejich závislosti na externím API systému OctoPrint.

Serverová část tak tvoří stabilní a flexibilní základ systému, který umožňuje bezpečnou správu uživatelů, efektivní přístup k datům a plynulou interakci s modulem pro analýzu obrazových dat.

5.4 Uživatelské rozhraní pro webovou aplikaci

Implementace uživatelského rozhraní aplikace se nachází ve složce `src`. Klíčovou složkou je `app`, která obsahuje podadresáře reprezentující jednotlivé stránky webu. Každá z těchto složek zpravidla obsahuje soubor `page.js`, jenž definuje JavaScriptový a HTML kód dané stránky. Dále může obsahovat i samostatný soubor se styly ve formátu `.css`, specifický pro danou část rozhraní.

Navigační lišta, která je zobrazována napříč celou aplikací, je implementována jako samostatná komponenta a do rozhraní je začleněna prostřednictvím souboru `layout.js`. Tento soubor slouží jako výchozí šablona rozvržení, skrze kterou se vykreslují všechny ostatní stránky. Zvolený přístup usnadňuje správu společných prvků a zajišťuje konzistentní vzhled uživatelského rozhraní.

Správa informací o aktuálně přihlášeném uživateli je řešena pomocí kontextu. Ve složce `context` se nachází implementace `UserContextu`, který zpřístupňuje údaje o jménu a oprávnění přihlášeného uživatele napříč celou aplikací. Díky tomu lze dynamicky přizpůsobovat zobrazované prvky rozhraní podle uživatelské role a zároveň zajistit, že se uživateli nebudou zobrazovat možnosti, k nimž nemá oprávnění.

Kapitola 6

Testování zásuvného modulu

V rámci této práce byl otestován vyvinutý plugin **Cvprinting** a porovnán s dvěma komerčně dostupnými řešeními: **Obico** a **OctoEverywhere**.

Testovací sestava zahrnovala Raspberry Pi 5 s kamerou Pi camera 3 a nainstalovaným systémem OctoPrint, který byl propojen s tiskárnou Creality Ender 3 V3 SE.

U služby Obico byla po registraci automaticky aktivována zkušební verze Pro, která umožňovala využít až 250 hodin detekce pomocí umělé inteligence. Služba OctoEverywhere byla testována ve své bezplatné verzi. Testování probíhalo za pomoci tisků se simulovanými chybami a tisků bez chyb.

6.1 Testování na simulovaných chybách

Pro účely testování bylo navrženo několik scénářů se simulovanými tiskovými chybami:

- **Test stringování** – Byly tištěny dvě věže umístěné v těsné blízkosti s cílem vyvolat stringování mezi nimi. Pro zvýšení pravděpodobnosti vzniku stringování bylo sníženo množství retrakce, zpomalena rychlost retrakce, zvýšena teplota trysky a rychlost tisku byla nastavena na 150 %. Věže byly orientovány tak, aby bylo vlákenní mezi nimi dobře viditelné na kameře.
- **Test stringování s natočením** – Tento test využíval stejný model a nastavení jako předchozí test, avšak model byl mírně natočen, aby byla viditelnost stringování na kameře omezena.
- **Test špaget** – Při tomto testu byl použit upravený G-code soubor, který způsobil tisk ve vzduchu, což vedlo ke vzniku špaget.
- **Test špaget s umístěným modelem** – Tento scénář navazoval na předchozí, ale na tiskovou podložku byl navíc umístěn hotový 3D model, aby se lépe simulovala situace, kdy k chybě dojde v průběhu běžného tisku.

Každý scénář byl testován s umístěním kamery na boční a přední části tiskárny. Umístění z boku poskytovalo nejlepší pohled na tištěný model, díky minimálnímu zakrytí modelu tiskovou hlavou. V případě pohledu zřepředu byl již model částečně zakryt tiskovou hlavou. Pohled shora nebyl testován, jelikož konstrukce tiskárny, na které se testy prováděly, by způsobila zakrytí značné části tiskové plochy.

Některé testy byly rovněž provedeny s různými typy filamentů, aby se ověřila schopnost modelu detekovat chyby u různých barev filamentů.

Pro každý tisk byl zaznamenán čas, kdy se tisková závada stala poprvé viditelnou na záznamu z kamery. Následně se sledovalo, zda a s jakým časovým odstupem od okamžiku, kdy byla chyba viditelná, jednotlivá řešení zaslala upozornění. K doručování upozornění byla ve všech případech využita platforma Discord.

V případě pluginu **Cvprinting** byly kromě času odeslání upozornění zaznamenávány také informace o míře jistoty detekce a detekované třídě chyby. Tyto údaje slouží k podrobnějšímu vyhodnocení přesnosti a konzistence detekčního modelu.

Nastavení jednotlivých řešení bylo následující:

- **Cvprinting** – minimální konfigurovaný práh pro odeslání upozornění byl nastaven na 40 %.
- **OctoEverywhere** – nastaveno tak, aby odesílalo upozornění při nejnižší úrovni jistoty.
- **Obico** – nastaveno na vysokou citlivost detekce.

Tento postup umožnil objektivní srovnání rychlosti a přesnosti detekce mezi jednotlivými řešeními jak u závad, tak v běžném tisku bez problémů.

Kamera z boku

Test stringování – Tisk trval celkem 9 minut. První výskyt závady byl patrný po 5 minutách. **Detekce:**

- Obico – bez upozornění
- OctoEverywhere – bez upozornění
- Cvprinting – upozornění po 1 minutě od výskytu závady (míra jistoty 42 %, třída: stringování)

Test stringování s natočením – Celková délka tisku byla 11 minut, závada se objevila po 5 minutách. **Detekce:**

- Obico – bez upozornění
- OctoEverywhere – upozornění po 5 minutách
- Cvprinting – upozornění po 6 minutách (míra jistoty 57 %, třída: stringování)

Test špaget – Tisk byl ukončen přibližně po 5 minutách, kdy všechny systémy zaslaly upozornění na chybu. Závada byla patrná okamžitě po začátku tisku. **Detekce:**

- Obico – upozornění po 5 minutách
- OctoEverywhere – upozornění po 2 minutách
- Cvprinting – upozornění po 2 minutách (míra jistoty 80 %, třída: špagety)

Test špaget s umístěným modelem – I v tomto případě byl tisk ukončen přibližně po 5 minutách, kdy všechny systémy zaslaly upozornění na chybu. Závada byla viditelná ihned po spuštění tisku. **Detekce:**

- Obico – upozornění po 4 minutách
- OctoEverywhere – upozornění po 1 minutě
- Cvprinting – upozornění po 1 minutě (míra jistoty 70 %, třída: špagety)

Dále byly provedeny dva testy za využití jiné barvy filamentu:

Test stringování – Tisk trval celkem 13 minut. První výskyt závady byl patrný po 5 minutách. **Detekce:**

- Obico – bez detekce
- OctoEverywhere – upozornění po 5 minutách
- Cvprinting – bez detekce

Test špaget – Tisk byl ukončen přibližně po 5 minutách, kdy všechny systémy zaslaly upozornění na chybu. Závada byla patrná okamžitě po spuštění tisku. **Detekce:**

- Obico – upozornění po 5 minutách
- OctoEverywhere – upozornění po 2 minutách
- Cvprinting – upozornění po 1 minutě (míra jistoty 43 %, třída: špagety)

Kamera zepředu

Test stringování – Tisk trval celkem 10 minut. První výskyt závady byl patrný po 5 minutách. **Detekce:**

- Obico – bez upozornění
- OctoEverywhere – bez upozornění
- Cvprinting – bez upozornění

Test špaget – Tisk byl ukončen přibližně po 7 minutách, kdy všechny systémy zaslaly upozornění na chybu. Závada byla patrná okamžitě po začátku tisku. **Detekce:**

- Obico – upozornění po 5 minutách
- OctoEverywhere – upozornění po 2 minutách
- Cvprinting – upozornění po 2 minutách (míra jistoty 73 %, třída: špagety)

Test špaget s umístěným modelem – Tisk byl ukončen přibližně po 6 minutách, kdy všechny systémy zaslaly upozornění na chybu. Závada byla patrná okamžitě po spuštění tisku. **Detekce:**

- Obico – upozornění po 5 minutách
- OctoEverywhere – upozornění po 2 minutách
- Cvprinting – upozornění po 2 minutách (míra jistoty 69 %, třída: špagety)

Test stringování s natočením – Celková délka tisku byla 11 minut. První výskyt závady byl patrný po 5 minutách. **Detekce:**

- Obico – bez upozornění
- OctoEverywhere – bez upozornění
- Cvprinting – bez upozornění

6.2 Testování na tisku bez chyby

Kromě testů simulujících chyby byly rovněž provedeny testy na úspěšně dokončených výtiscích, aby se ověřilo chování systémů detekce v běžných podmínkách bez výskytu závad. V těchto případech nebyly použity žádné specifické scénáře – tisk probíhal standardním způsobem s validním G-code souborem. Testy v této kategorii byly prováděny pouze s kamerou umístěnou z boku.

Testování bylo provedeno na více než deseti hodinách tisků, přičemž bylo zahrnuto více než pět různých výtisků. Během jednoho tisku o délce přibližně 2,5 hodiny došlo k pěti falešně pozitivním detekcím, všechny tyto detekce byly pro třídu stringování. Hodnota spolehlivosti těchto detekcí se pohybovala v rozmezí od 41 % do 56 %. Systémy OctoEverywhere a Obico během těchto tisků nezaslaly žádné upozornění na falešně pozitivní detekci. Pro zvýšení reprezentativnosti výsledků byly testy prováděny s různými typy filamentů, zahrnujícími rozdílné barvy i materiály.

Kapitola 7

Závěr

Cílem této práce bylo vytvořit systém využívající počítačové vidění pro detekci nejběžnějších chyb během 3D tisku a uživatelské rozhraní, které bude možné použít jak při provozu jedné tiskárny, tak i při správě více tiskáren, například na fakultě.

Systém byl navržen tak, aby byl snadno nasaditelný. Byl implementován jako zásuvný modul pro aplikaci OctoPrint a plně běží na zařízení Raspberry Pi 5, bez nutnosti připojování dalších zařízení pro vyhodnocování obrazu. Modul zároveň obsahuje jednoduché uživatelské rozhraní pro správu jednotlivé tiskárny.

Pro správu více tiskáren byla vytvořena samostatná webová aplikace umožňující sledování stavu tisku na jednotlivých tiskárnách včetně výsledků detekce chyb. Aplikace rovněž umožňuje nastavení parametrů detekce, jako je práh jistoty, a správu oprávnění uživatelů pro přístup k jednotlivým tiskárnám.

Součástí práce bylo také vytvoření datové sady obsahující snímky s anotacemi pěti různých tříd chyb. Tato datová sada sloužila k doladění předtrénovaného modelu počítačového vidění YOLOv11s.

Výsledky ukazují, že je systém schopen detekovat chyby při tisku, jako je stringování nebo tvorba tzv. špaget, v reálném čase i na relativně omezeném hardwaru. V případě detekce chyby je uživatel upozorněn a na základě nastavení může být tisk automaticky pozastaven.

Zásuvný modul byl uveřejněn 21. dubna pomocí správce zásuvných modulů pro OctoPrint. Během procesu zveřejnění jsem obdržel zpětnou vazbu ke kódu, díky které jsem opravil kód sloužící pro kontrolu aktualizací a také opravil způsob přístupu k složkám pro ukládání dat.

Datovou sadu je možné dále rozšiřovat, zejména v kategoriích, jako je přeextrudování, podextrudování a tvorba mezer, které při trénování vykazovaly vyšší míru chybovosti a nebyly proto ve výsledném modelu plně využity. Vylepšená datová sada by mohla sloužit jako základ pro modely schopné detekovat širší spektrum chyb, čímž by došlo k dalšímu rozšíření funkcionality systému. Dalšími možnostmi rozvoje jsou podpora více platforem pro upozornění nebo možnost zobrazení ohraničujících boxů přímo v zaslaných upozorněních.

Literatura

- [1] *Complete 3D printing troubleshooting guide. All common 3D printing problems and their solutions. Includes both FDM and SLA 3D printing issues.* online. ALL3DP. Dostupné z: <https://all3dp.com/1/common-3d-printing-problems-troubleshooting-3d-printer-issues/#section-fdm-3d-printing-problems-nothing-is-printing>. [cit. 2025-04-12].
- [2] *GlobalInterpreterLock* online. Python.org. Dostupné z: <https://wiki.python.org/moin/GlobalInterpreterLock>. [cit. 2025-04-18].
- [3] *Image segmentation detailed overview [Updated 2024]* online. SuperAnnotate. Dostupné z: <https://www.superannotate.com/blog/image-segmentation-for-machine-learning>. [cit. 2025-04-15].
- [4] *Print Quality Troubleshooting Guide* online. Simplify3D. Dostupné z: <https://www.simplify3d.com/resources/print-quality-troubleshooting/>. [cit. 2025-04-12].
- [5] *Spaghetti monster* online. Prusa Research. Dostupné z: https://help.prusa3d.com/article/spaghetti-monster_1999. [cit. 2025-04-12].
- [6] BAUMANN, A.; APPAVOO, J.; KRIEGER, O. et al. *A fork() in the road*. 2019. Dostupné z: <https://web.eecs.umich.edu/~ryanph/jhu/cs318/fall122/readings/fork-hotos19.pdf>.
- [7] BRION, D. A. J. a PATTINSON, S. W. Generalisable 3D printing error detection and correction via multi-head neural networks. *Nature Communications*. Nature Publishing Group, 2022, sv. 13, č. 1, s. 4654.
- [8] CSURKA, G.; VOLPI, R. a CHIDLOVSKII, B. Semantic Image Segmentation: Two Decades of Research. *Foundations and Trends in Computer Graphics and Vision*. Now Publishers, 2023, sv. 14, 1–2, s. 1–175.
- [9] ELGENDY, M. *Deep Learning for Vision Systems*. Manning, 2020. ISBN 9781617296192.
- [10] GIBSON, I.; ROSEN, D.; STUCKER, B. et al. *Additive Manufacturing Technologies*. 3rd. Cham, Switzerland: Springer, 2021. ISBN 978-3-030-56127-7.
- [11] GOODFELLOW, I.; BENGIO, Y. a COURVILLE, A. *Deep Learning*. MIT Press, 2016. ISBN 9780262035613.

- [12] HEINZ, M. *Počítačové vidění pro sledování 3D tisku*. Brno, CZ, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: <https://www.vut.cz/studenti/zav-prace/detail/145063>.
- [13] JADHAV, Y.; PAK, P. a FARIMANI, A. B. LLM-3D Print: Large Language Models To Monitor and Control 3D Printing. *ArXiv preprint arXiv:2408.14307*, 2024. Dostupné z: <https://arxiv.org/abs/2408.14307>.
- [14] KLOSKI, L. W. a KLOSKI, N. *Getting Started with 3D Printing, 2nd Edition*. 2. vyd. Make: Community, 2021. ISBN 9781680456431.
- [15] MODERTATOR_. *Very light stringing?* online. Dostupné z: https://www.reddit.com/r/FixMyPrint/comments/on6ww/very_light_stringing/. [cit. 2025-04-12].
- [16] NIELSEN, M. A. *Neural Networks and Deep Learning* online. Determination Press, 2015. Dostupné z: <http://neuralnetworksanddeeplearning.com/>.
- [17] OCTOPRINT. *Octoprint* online. Dostupné z: <https://octoprint.org/>. [cit. 2025-04-12].
- [18] *Řešení potíží při tisku* online. Prusa Research. Dostupné z: https://help.prusa3d.com/cs/category/reseni-potizi-pri-tisku_225. [cit. 2025-04-12].
- [19] ROSEBROCK, A. *Deep Learning for Computer Vision with Python*. 1. vyd. PYIMAGESEARCH, 2017. ISBN 9781722487836.
- [20] SANI, A. R.; ZOLFAGHARIAN, A. a KOUZANI, A. Z. Automated defects detection in extrusion 3D printing using YOLO models. *Journal of Intelligent Manufacturing*. Springer, 2024.
- [21] SZELISKI, R. *Computer Vision: Algorithms and Applications*. 2. vyd. Springer Nature, 2022. ISBN 3030343723.
- [22] ULTRALYTICS. *Convolutional Neural Network (CNN)* online. Dostupné z: <https://www.ultralytics.com/glossary/convolutional-neural-network-cnn>. [cit. 2025-04-10].
- [23] ZHANG, H.; ZONG, Z.; YAO, Y. et al. Multi-Axis 3D Printing Defect Detecting by Machine Vision with Convolutional Neural Networks. *Experimental Techniques*. Springer, 2023, sv. 47, č. 5, s. 619–631.

Příloha A

Plakát

Počítačové vidění pro sledování 3D tisku

Viliam Chudáčik • xchuda06@stud.fit.vutbr.cz
Vedoucí: doc. RNDr. Pavel Šmrž, Ph.D. • FIT VUT v Brně • 2024/2025

Úvod a motivace

3D tisk je cenově dostupný a populární, ale náchylný na řadu chyb. Např. „špagety“ vznikají při selhání adheze první vrstvy nebo stringování, které se projevuje zanecháváním tenkých vláken plastu při tisku, podobně jako tomu je při použití tavných pistolí. Tyto chyby lze sice detekovat vizuálně uživatelem, ale u dlouhotrvajících tisků to není efektivní. Proto se nabízí řešení využívající počítačové vidění pro detekci těchto chyb.

Cíl a přístup

Systém byl navržen s důrazem na jednoduchou instalaci a použití.

- ▶ Detekce chyb pomocí předtrénovaného modelu YOLOv11s
- ▶ MNN framework pro rychlé zpracování snímku
- ▶ běh celého systému na Raspberry Pi 5
- ▶ integrace do OctoPrint formou zásuvného modulu (pluginu)
- ▶ notifikace uživateli při výskytu chyby pomocí Discordu a Telegramu
- ▶ možnost pozastavení tisku při detekci chyby s dostatečnou mírou jistoty.

Architektura systému

Užiti webkamery připojené k OctoPrintu

Detekce každých 5 vteřin, pro akci nutné alespoň dvě chyby po sobě

Analýza v samostatném procesu, užítí fronty pro odesílání zpráv o detekci

Volitelná uživatelská aplikace pro správu více tiskáren z jednoho uživatelského rozhraní

Výsledky

- ▶ Detekce probíhá v reálném čase
- ▶ Detekce chyb typu špagety a stringování
- ▶ Výsledky při testování detekce na podobné úrovni jako komerční řešení OctoEverywhere a Obico
- ▶ Nízká míra falešně pozitivních reakcí
- ▶ Projekt je dostupný na GitHubu a v oficiálním správci zásuvných modulů pro OctoPrint pod názvem OctoPrint-CVPrinting.

Shrnutí

Navržený systém je snadno rozšiřitelný a vhodný jak pro domácí užítí tak i pro využití například na fakultě za použití volitelného uživatelského rozhraní. Systém spolehlivě rozpoznává chyby, čímž umožňuje šetřit materiál a čas.



Tento projekt byl realizován v rámci bakalářské práce na FIT VUT v Brně.

Obrázek A.1: Plakát