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

STROJOVÉ UČENÍ ZE SYNTETICKÉ DATOVÉ SADY PRO POČÍTÁNÍ PŘEPRAVEK V OBRAZE

MACHINE LEARNING ON SYNTHETIC DATA FOR COUNTING CRATES IN IMAGES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ONDŘEJ KOĎOUSEK**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

154402

Ústav: Ústav počítačové grafiky a multimédií (UPGM)
Student: **Koďousek Ondřej**
Program: Informační technologie
Název: **Strojové učení ze syntetické datové sady pro počítání přepravek v obraze**
Kategorie: Zpracování obrazu
Akademický rok: 2023/24

Zadání:

1. Seznamte se s problematikou detekce objektů v kontextu počítačového vidění.
2. Vyhledejte a prostudujte existující detektory objektů založené na konvolučních neuronových sítích.
3. Získejte, shromážděte či generujte vhodné datové sady obsahující přepravky – konzultujte s vedoucím.
4. Experimentujte s dostupnými detektory objektů nad datovými sadami, vyberte vhodný detektor a jeho režim použití.
5. Navrhněte a implementujte aplikaci pro počítání přepravek ve scéně zabírané statickou kamerou.
6. Demonstrujte funkčnost vytvořeného řešení a diskutujte jeho omezení a možnosti jejich překonání.
7. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Richard Szeliski: Computer Vision: Algorithms and Applications, Springer, 2011
- Goodfellow, Bengio, Courville: Deep Learning, MIT Press, 2016
- Andriy Burkov: The hundred-page machine learning book, Large Print Book, 2019
- Bharath Ramsundar, Reza Bosagh Zadeh: TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning, O'Reily Media, 2018
- Vincent Dumoulin, Francesco Visin: A guide to convolution arithmetic for deep learning, https://arxiv.org/abs/1603.07285
- Gary Bradski, Adrian Kaehler: Learning OpenCV; Computer Vision with the OpenCV Library, O'Reilly Media, 2008

Při obhajobě semestrální části projektu je požadováno:
body 1. a 2., značné rozpracování bodů 3. a 4.

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

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, prof. Dr. Ing.
Datum zadání: 1.11.2023
Termín pro odevzdání: 9.5.2024
Datum schválení: 9.11.2023

Abstrakt

Cílem práce je vytvořit proces, který spočítá kolik je ve videozáznamu či statickém snímku přepravek. Toho se dosáhne pomocí modelu, který je natrénován na syntetické datové sadě, a následně jsou výsledky upraveny na úrovni individuálních snímků a dále na úrovni souvislého videozáznamu. Tahle syntetická datový sada je vygenerována pomocí skriptu v programu Blender s využitím Octane Render, pro vyšší úroveň fotorealismu. Přínos úspěšného trénování na syntetické datové sadě je rychlejší a především automatické vytváření anotací. Jelikož se anotace generují se samotným snímkem, není problém vygenerovat velké množství snímku bez jediné manuální anotace. Dalším přínosem je náskok při vytváření modelu na detekci objektů, které jsou na trhu nové a nemají dostatek dat, případně jsou teprve ve výrobě. Detekoval jsem na statických snímkcích i ve videozáznamu, u obou případů jsem s modelem natrénovaným na syntetických datech dosahoval úspěšnosti nad 90 %.

Abstract

The goal of this work is to create a process that counts how many crates are in a video or still image. This is done by using a model that is trained on a synthetic dataset, and then the results are adjusted at the individual frame level and then at the continuous video frame level. This synthetic dataset is generated using a script in Blender using Octane Render, for a higher level of photorealism. The benefit of successfully training on the synthetic dataset is faster and especially automatic annotation. Since the annotations are generated with the image itself, it is not a problem to generate a large number of images without a single manual annotation. Another benefit is a head start in model generation for detecting objects that are new to the market and lack sufficient data, or are only in production. I have detected in still images and video, and in both cases I achieved success rates above 90 % with a model trained on synthetic data.

Klíčová slova

strojové učení, syntetická datová sada, počítání přepravek, CenterNet, Blender, Octane Engine

Keywords

machine learning, synthetic dataset, counting crates, CenterNet, Blender, Octane Engine

Citace

KOĎOUSEK, Ondřej. *Strojové učení ze syntetické datové sady pro počítání přepravek v obraze*. Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Strojové učení ze syntetické datové sady pro počítání přepravek v obraze

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. Ing. Adama Herouta, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Ondřej Koďousek
8. května 2024

Poděkování

Chtěl bych poděkovat vedoucímu mé práce, prof. Ing. Adamovi Heroutovi, Ph.D., za přesměrování mé původní práce na mnohem více zajímavější téma o syntetických datech, za odbornou pomoc při samotné tvorbě a také udělení přístupu na školní server Sophie.

Obsah

Kapitola 1

Úvod

Cílem práce je vytvořit proces, který dokáže spolehlivě detekovat a spočítat přepravky v obraze. Tato práce toho dosahuje pomocí detekce objektů konvoluční neuronovou sítí CenterNet, s tím, že práce je zaměřena na učení se na syntetických datech, která sám generuji. Detekce je zaměřena pouze na jednotlivé viditelné přepravky, v případě převozu na paletách, kde zadní přepravky nelze vidět, se neberou v potaz.

Trénování na syntetických datech může být obtížné, jelikož detektor má za úkol predikovat objekty z reálného světa, i přes to, že mezera mezi reálnými a syntetickými snímky může být značná, proto je důležitý důraz na fotorealismus a co nejpřesnější prezentaci reality. Každopádně za předpokladu, že tento postup má dostatečnou úspěšnost, dokáže nabídnout několik značných vylepšení pro počítačové učení. Jednou z výhod je zcela automatická anotace a dále možnost získání dat o objektech, která nemají velké množství reálných dat k dispozici.

Můj skript, který využívá program Blender spolu s Octane Render, dokáže vytvořit scénu s náhodnými parametry, jako je světlo, barva a textura přepravek, pozice kamery a podobně. I přesto že se jedná o scénu s náhodnými parametry, je známo, kde se přepravky vygenerují, a právě díky tomu lze pomocí překladu 3D souřadnic na 2D souřadnice ve výsledném snímku vyhodnotit, kde vytvořit anotaci pro každou přepravku. Tedy zjednodušeně skript vytvoří scénu, vykreslí snímek a zároveň uloží anotaci. Tento proces je zcela automatický, stačí tedy, aby znalý uživatel zadal požadované parametry, jako je například velikost datové sady, tedy počet trénovacích, validačních a testovacích snímků, rozlišení, typ přepravky, parametry ovlivňující množství přepravek a variaci, a následně samotné parametry scény, jako je výška kamery, průhlednost fólií, intenzita a variace denního světla a podobně. Po tomto procesu je celá datová sada ihned připravena na trénování v neuronové síti CenterNet.

Po natrénování modelu následuje proces detekce a spočítání přepravek. Po vytvoření predikcí se dále vyhodnocuje, jestli se někde nedetekovala přepravka navíc, nebo naopak, jestli model nepřehlédl přepravku tam, kde by pravděpodobně měla být, například když je sloupec s predikcemi, mezi kterými je podezřelá mezera. Jako výslednou hodnotou se bere nejdůvěryhodnější počet přepravek, který byl detekován v daném časovém úseku, finální detekce je tedy na videozáznamu, ale lze detekovat i na statických snímcích. Na poskytnutém záznamu se přepravují přepravky buď z nákladního auta do skladu nebo naopak. Sbírají se snímky, které spolu souvisí, tedy když se detekuje změna směru, nebo pokud přepravky zmizí, tak se ukončí sběr snímků. Tyto snímky se různými algoritmy vyfiltrují, a nejčastější počet přepravek ze všech snímků je bráno jako výsledný počet přepravek.

Kapitola 2

Cíl práce a problematika syntetických dat

Proces začíná videozáznamem kamerového systému na vstupu do skladu (viz obrázek 2.1). Na něm se pohybují zaměstnanci firmy a přepravují prázdné přepravky z nakládacího auta do budovy nebo naopak. Nákladní automobil stojí na jednom místě, a kamera vidí do otevřeného prostoru a částečně do jedné místnosti. Přepravky mohou být buď přeneseny do této místnosti nebo přivezeny do otevřeného prostoru, či naopak. Program musí podporovat tyto různé scénáře a umožnit uživateli vybrat si vhodné nastavení. Zároveň je celá oblast venku, proto mohou být světelné podmínky značně odlišné v závislosti na denní době.

Přepravky E1 a E2 jsou obvykle přepravovány na paletách, přičemž na jedné paletě je vždy jen jeden typ přepravky. Mohou být do jisté míry opotřebeny, příležitostně mají větší otvory, degradovanou barvu a občas jsou na nich různě umístěné nálepky. Přepravky E1 a E2 jsou přepravky evropského standardu pro přepravu masných výrobků. Jejich vnější rozměry jsou $600 \times 400 \times 125$mm pro E1 a $600 \times 400 \times 200$mm pro E2. E1 má na přední straně jeden obdélníkový otvor a dva čtvercové otvory, zatímco E2 má navíc dva čtvercové otvory a více vyztuženou horní část.

Při detekci přepravek se nedetekuje celá přepravka, ale zaměřuje se právě na tyto otvory. Jelikož má přepravka E2 více otvorů a tedy i více identifikačních prvků, na kterých se neuronová síť může lépe naučit, tak je předpoklad, že úspěšnost detekci E2 bude větší než E1. Zároveň jelikož E1 je v podstatě podmnožinou E2, tak očekávám více případů, kdy se přepravka E2 nesprávně vyhodnotí jako E1.

Dále jsou některé části palety obalené několika vrstvami průhledné fólie, jak lze vidět na obrázku 2.1. Tato fólie slouží k udržení přepravek pohromadě a je nanášena ručně, proto je třeba mít vysokou míru tolerance pro spolehlivou detekci, jelikož počet vrstev a jejich pozice nejsou pevně dané. Za určitých úhlů může být odlesk z fólie tak silný, že přepravky jsou obtížně rozpoznatelné i lidským okem.

Záznam má rovněž vysokou míru JPEG komprese. Místa 16×16 nebo 8×8 pixelů jsou často různě posunuté, hodnoty jednotlivých pixelů mohou být zpřeházené, občas je celá tato plocha z jediné barvy a podobné artefakty, které vznikají vysokou mírou komprese.

V obrázku 2.2 jsou přiblíženy 2 oblasti. V horní oblasti lze vidět, že otvory přepravky pokryté fólií jsou špatně rozeznatelné i pro lidské oko. Zároveň lze vidět značné artefakty způsobeny kompresí, pixely vypadají jako kdyby byli seskupeny do skupiny 1×8. V dolní oblasti jsou viditelné podobné artefakty, s občasnými artefakty v seskupení 8×8. Tyto

Obrázek 2.1: Příklad vstupu. Zde lze vidět odrazy fólií, degradovanou barvu přepravek, nálepky na různých pozicích. I bez značného přiblížení lze pozorovat různé artefakty způsobené kompresí.

vlastnosti ztěžují detekci, a bude potřeba je u syntetických dat replikovat, pravděpodobně jednoduchou augmentací.

2.1 Generování syntetické datové sady pomocí CGI

Tato metoda je zaměření této práce. Jedná se o vytvoření 3D scény a vykreslení výsledných snímků pomocí CGI (Computer-Generated Imagery). Je zapotřebí vytvořit důvěryhodné 3D modely objektů, fotorealistické světelné podmínky a důvěryhodně vypadající materiál. Samozřejmě jsou i další parametry potřebné k tomu, aby výsledný obrázek vypadal fotorealisticky, a měl tedy šanci úspěšně natrénovat neuronovou síť.

Jelikož je scéna plně pod kontrolou, tak jsou známy 3D souřadnice všech objektů a také parametry kamery, pomocí těchto elementů lze vypočítat 2D bounding box souřadnice pomocí perspektivní projekce[12] a zjistit tak polohu anotace na výsledném snímku, a tím zautomatizovat proces vytváření anotací.

Problémem této metody je právě vytvoření důvěryhodného snímku, který je zároveň vygenerován s nějakými náhodnými odchylkami. Nedává smysl ručně vytvářet každou scénu, proto je zapotřebí generovat scény s náhodnými elementy, jako je osvětlení a úhel kamery. Každopádně tohle je opět problém, jelikož není přesně známo, jak moc se lze lišit od originálu, a také není známo, co přesně lze považovat za originál. Pokud je snaha vytvořit scénu, která připomíná jediný reálný snímek, tak by se jednalo o velmi specifický přístup, a model by měl problém detekovat stejné objekty v jiných podmínkách.

Proto jsem vytvořil parametry scény (ve zbytku práce budu tyhle specifické parametry nazývat parametry scény, aby se výrazy nepletli s ostatními parametry), jako je například poloha kamery, intenzita a variace osvětlení, a nebo vlastnosti materiálu. Plán je prozkou-

Obrázek 2.2: Detail vstupního snímku s přiblíženými oblastmi snímku. V horním přiblížení nelze bez kontextu téměř vůbec rozeznat, o jaký objekt se jedná. V dolní části lze vidět části 8×8 či 8×1 pixelů, které se spojují v jeden celek a vytvářejí různé artefakty.

mávat výsledky při změně těchto parametrů scény, a tím se co nejvíce přiblížit k nejlepší možné úspěšnosti.

Je také důležité zaručit, aby nebyl celý bounding box zakrytý jiným objektem, nebo aby osvětlení nebylo tak slabé, aby byl snímek efektivně černá barva, nebo naopak nereálně intenzivní světlo, případně nesprávné chování materiálu a jiné podmínky, které neodpovídají realitě. Tyto podmínky se sčítají a vytvářejí čím dál více komplexnější problém.

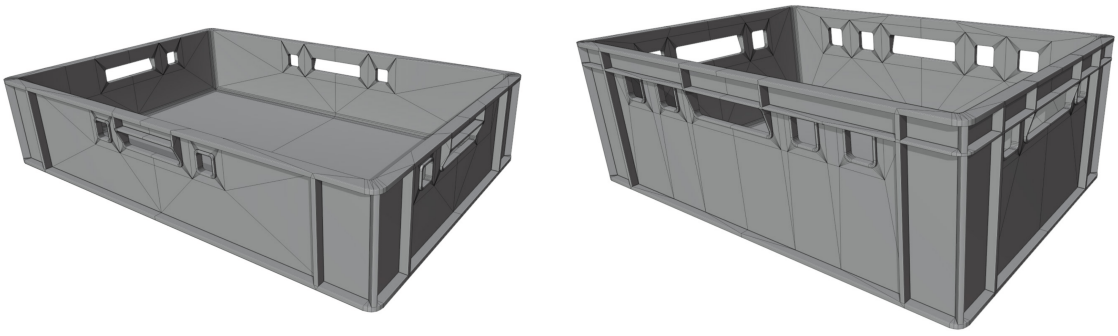
Je možnost využít syntetická data právě k zacelení nedostatků reálné sady a tenhle přístup se často využívá, každopádně tohle není přímo cílem práce, jako jeden z cílů chci vytvořit dostatečně úspěšný model, který byl vytvořen pouze ze syntetických snímků. Samozřejmě hlavním cílem je co nejlépe spočítat přepravky, bez ohledu na postup, ale myslím si, že tyto vedlejší cíle mohou mít velmi zajímavé výsledky.

2.1.1 BlenderAPI skript a generování scény

Hlavní část generování datové sady se odehrává v programu Blender[1]. Blender je bezplatný, multiplatformní, a zároveň má skvělou podporu a dokumentaci pro skriptování pomocí Blender API[2]. Kvůli těmito vlastnostem je Blender pro mé účely ideální.

BlenderAPI se často používá pro vytváření různých rozšíření, nebo obecně automatizace pouze části práce, každopádně lze vytvořit dostatečně sofistikovaný skript, který se spustí v režimu bez rozhraní (headless), importuje objekty, vytvoří scénu a správně nastaví její parametry, vytvoří co je potřeba, a nakonec vykreslí výsledné snímky. Takhle funguje můj skript, není potřeba žádného zásahu, a stačí základní znalosti pro jeho spuštění, a tedy i pro vytvoření samotné datové sady.

[1] Blender: www.blender.org
[2] Dokumentace BlenderAPI: docs.blender.org/api/current



(a) Přepravka E1. (b) Přepravka E2.

Obrázek 2.3: Ukázky vytvořených modelů přepravek E1 a E2. Anotace se vytváří na přední části s otvory.

Skript importuje modely které jsem vymodeloval (obrázky 2.3a a 2.3b), ale skript není moc flexibilní na jiné objekty, nejedná se o robustní a všestranný framework. Pro přidání nového druhu přepravky či objektu je potřeba správně modifikovat kód. V případě implementace dalších přepravek či jiných jednoduchých objektů ve tvaru krychle nebo hranolu by skript měl správně fungovat, když se adekvátně změní rozměry v kódu. V případě složitějších objektů by tohle bylo více obtížné, a bylo by vhodné změnit mechanismus na nastavení pozice objektů, jelikož ne všechny objekty se skladují v řádcích a sloupcích. Také by bylo zapotřebí změnit pozici, ve které se vytvoří anotace daného objektu.

2.1.2 Kubric a pohled na existující alternativy

Kubric[9] je známý projekt, který dokáže na vyžádání generovat syntetickou datovou sadu. Využívá Blender, ačkoliv v současnosti je omezen na dosti zastaralou verzi 2.93. Je možné vytvořit simulace pomocí PyBullet[3], a tím se model může lépe učit na modelech pod více úhly. Kubric zároveň vytváří informace o hloubce, souřadnicích objektů a obdobných dodatečných datech. Využívá výchozí vykreslovací engine od Blenderu, který se jmenuje Cycles. Kubric podporuje ShapeNet[4], takže je možné využívat již anotované 3D objekty přímo z této služby.

Kubric používá animace a fyzické simulace na vytvoření variace snímků, zatímco mé řešení generuje statické scény, a pro každý snímek generuje jinou, unikátní scénu s jinými vlastnostmi, elementy, a úhly kamery. Dále je projekt Kubric zaměřený na vytváření syntetických snímků pro jakékoliv objekty, zatímco má práce je specializovaná na detekci daných přepravek.

Přístup Kubric a můj přístup jsou značně odlišné, a bylo by proto obtížné je porovnávat, jelikož rozdílné výsledky by byly ovlivněné nejen jiným přístupem k vytváření scény, ale zároveň i jiným použitým vykreslovacího engine.

2.1.3 Octane Render a fotorealismus

Octane Render[5] je vykreslovací engine, který je nezkreslený (unbiased), tedy ve velké míře nevyužívá umělé efekty a snaží se co nejvíce reprodukovat fyzikální chování světla a materi-

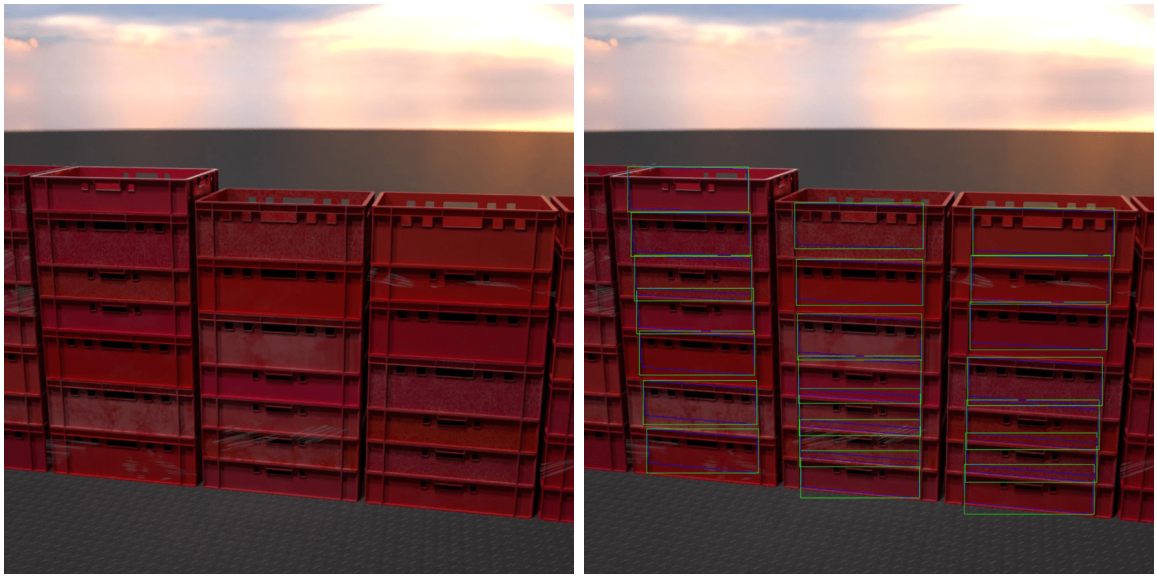
[3]Fyzikální simulátor PyBullet: pybullet.org
[4]ShapeNet, rozsáhlá databáze 3D modelů pro počítačové vidění a strojové učení: shapenet.org
[5]Vykreslovací engine Octane Render: home.otoy.com/render/octane-render

Obrázek 2.4: Ukázky snímků vykreslených mým skriptem. Rozlišení jednotlivých snímků je 1024×1024 pixelů, v mých datových sadách jsem téměř vždy používal menší rozlišení, a to 512×512 pixelů. Bílé objekty mají za úkol napodobit odrazy fólie.

álu, zároveň může využívat stopování cest (path tracing) a další pokročilé metody. Vykreslování probíhá na grafické kartě, a program je známý právě kvůli své rychlosti k poměru s fotorealismem i při výchozím nastavení. Jeho nevýhodami je menší robustnost oproti některým konkurenčním programů. Například konkurenční Arnold[6] je pomalejší, využívá procesor a je obtížnější pro začátečníky vytvořit realisticky vypadající snímky, každopádně při větších projektech, jako jsou například delší animace s více scénami, je často vhodnější volbou.

Každopádně tohle neplatí v mém případě. Pro mé účely je Octane dobrou volbou, jelikož je velmi jednoduché vytvořit realistické snímky, a již mám s programem zkušenosti. I základní osvětlení vypadá s výchozím nastavením věrohodně, a to je zejména důležité, jelikož skript potřebuje vytvořit tisíce snímků, každý s náhodně vytvořeným osvětlením,

[6] Vykreslovací engine Autodesk Arnold Renderer: https://www.autodesk.com/products/arnold/



Obrázek 2.5: Příklad snímku vykresleného mým skriptem. Pravý obrázek má vyznačené anotace, tedy bounding box pomocí zelené barvy, a segmentační masku pomocí modré barvy.

a všechny musí vypadat fotorealisticky. U ostatních vykreslovacích engine tohle může být značně obtížnější, jak jsem se přesvědčil při práci s Eevee v další kapitole 2.1.4.

Koláž s několika výslednými snímky si můžete prohlédnout na obrázku 2.4, a na obrázku 2.5 můžete vidět obrázek, kde je vizualizován bounding box a také segmentační maska.

Octane Render je placený, každopádně v případě jeho implementace v Blenderu nabízí velmi štědrou neplacenou verzi, která nemá značné limitace. Kvůli těmito vlastnostem, a hlavně kvůli mé předešlé zkušenosti, jsem si zvolil právě Octane Render.

2.1.4 Nezdařené pokusy s Eevee

Vyzkoušel jsem i vykreslovací engine Eevee[7], který je součástí Blenderu od verze 3.0. Je známy pro vykreslování snímků v reálném čase, každopádně pro mé účely nebyl dostatečně fotorealistický a kvůli potřebě vytvoření zcela nové scény pro každý snímek nemá rychlejší vykreslování tak značný přínos. V případě změny principu generování scény, konkrétně kdyby se generovala nové scéna každých N snímků místo u všech snímků, a každý snímek by měl jiný úhel pohledu, tak by Eevee byla více prospěšná a dokázala by být vícenásobně rychlejší.

Každopádně byl problém s vytvořením důvěryhodného světla, které by bylo možno náhodně měnit. Úspěšnost výsledného modelu byla velmi nízká, a jedinou výhodou bylo zhruba dvakrát méně času potřebných na vytvoření snímků, jelikož vygenerování scény v mém skriptu trvá zhruba 40 % až 50 % celkové doby na vytvoření snímku.

Největším problémem bylo vytvořit přirozené světlo uvnitř přepravky, jak lze vidět na obrázku 2.6. Vnitřek přepravky téměř nijak neodráží světlo, a kvůli tomu je mnohem více tmavější než by ve skutečnosti měl být. To je pravděpodobně následkem toho, že Eevee je optimalizovaná na vykreslování v reálném čase, a kvůli toho nepoužívá pokročilejší a více realistické metody, které takové problémy nemají. Komplikovaným způsobem se mi povedlo

[7]Vykreslovací engine Eevee: docs.blender.org/manual/en/latest/render/eevee

Obrázek 2.6: Příklad snímku vykreslený pomocí Eevee. Lze vidět velmi nepřirozené osvětlení, uměle vypadající materiál, a celkově se jedná o velmi nepřirozeně vypadající snímek.

zvýšit jasnost uvnitř přepravek při jistých světelných podmínkách, ale to zároveň způsobilo horší výsledky při odlišných podmínkách. Jelikož bylo obtížné vytvořit přijatelné podmínky manuálně, bylo by ještě obtížnější vytvořit tyhle podmínky automaticky a spolehlivě s využitím náhodné generace.

Kvůli těmto důvodům jsem usoudil, že tenhle přístup nikam nevede a přešel jsem zpátky na Octane. Tím nechci odrazovat od snahy vytváření syntetických dat pomocích podobných engine vykreslující v reálném čase, pouze mé pokusy s Eevee nic použitelného nevytvořily.

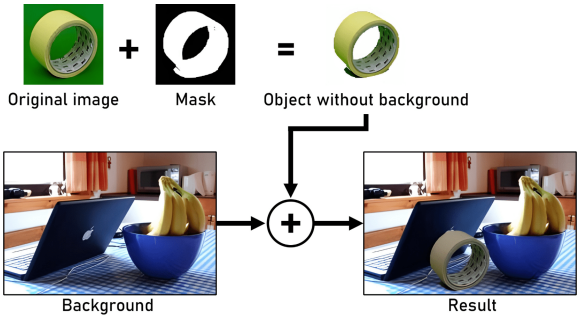
2.1.5 Technické problémy s Octane

Při vývoji jsem se potkal s několika problémy ohledně Octane, které se mi nepovedlo vyřešit. Tím největším je, že Octane, a tedy i celý program na vytváření syntetických dat, nelze rozumně zprovoznit na Linuxu, a lze ho tedy použít pouze na opečačním systému Windows.

Podařilo se mi program zprovoznit na mém lokálním Ubuntu 22.4, i na školním serveru, každopádně v obou případech trvá vykreslení snímku nepřirozeně dlouho. Takže program sice funguje, ale vykreslení snímku, které by mělo trvat asi 15 sekund, trvá téměř 2 minuty. Tenhle problém se mi nepodařilo vyřešit, a zdá se, že i přestože Octane podporuje Linux, tak naprostá většina uživatelů program využívá na systému Windows, proto se mi povedlo najít jen pár lidí se zhruba podobným problémem, ale řešení jsem nenalezl.

Dále je problém s novějšími verzemi, a to i na Windows. Novější verze Octane, které pomalu přecházejí na Blender 4.0, mají dle mé zkušenosti problém s referencemi různých atributů. Tedy Octane lze běžně využít pomocí rozhraní, ale při skriptování pomocí Blender API je obtížné program ovládat, protože příkazy, které fungují v dřívější verzi, se odkazují na neexistující objekty. Tohle nelze vyřešit jednoduchou úpravou kódu, jelikož některé objekty jednoduše nebyly správně aktualizovány v nových verzích a nelze je tedy nijak ovládat pomocí skriptu. Proto jsem zůstal u Octane verze 27.15.

Dalším problémem je, že Octane přestane odpovídat po nějaké době. Dle mého odhadu se jedná o problém s únikem paměti. Proto jsem vytvořil krátký skript, který spouští hlavní



Obrázek 2.7: Ukázka postupu vytvoření snímku pomocí kompozice, pomocí snímku který slouží jako pozadí, a snímku s požadovavým objektem. Převzato z [2], a dále upraveno.

skript, ale také hlídacího psa procesu. Tento proces periodicky kontroluje, jestli skript vytvořil snímek. Pokud ne, ukončí proces Octane a restartuje skript, který dále pokračuje od posledního snímku. Ve výsledku tedy program běží spolehlivě, občas se ale zdrží tím, že se musí restartovat.

2.2 Ostatní metody vytváření syntetických dat

2.2.1 Generování syntetické datové sady pomocí kombinování a míchání snímků

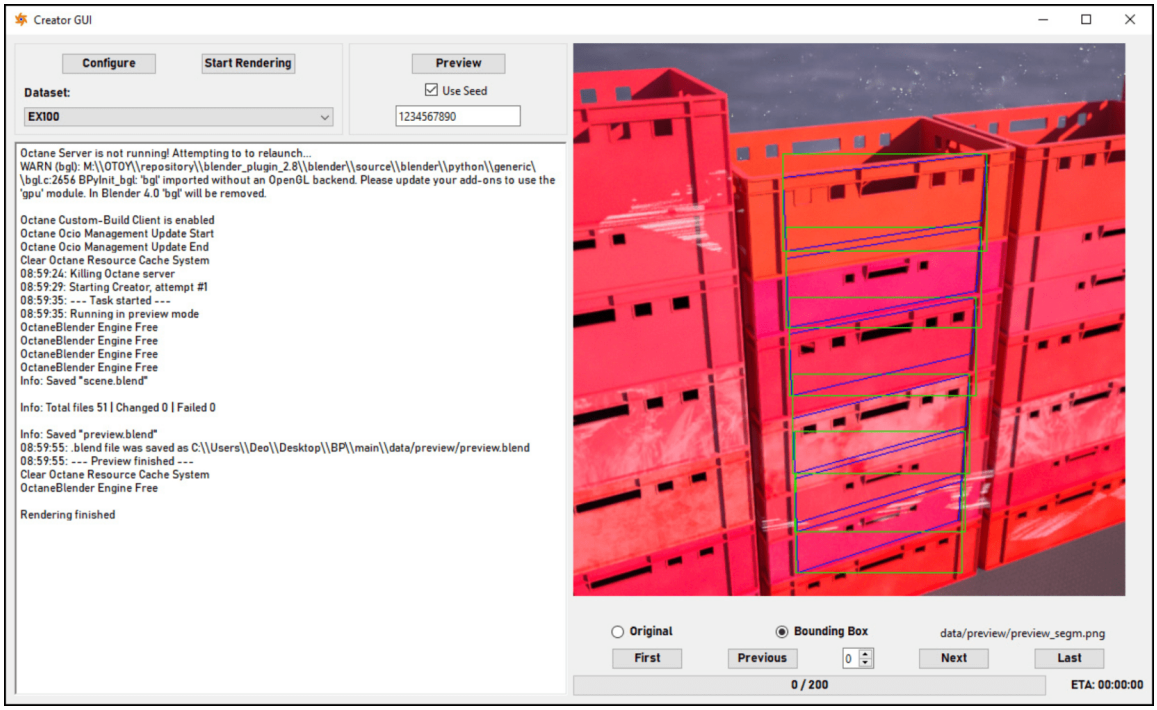
Jednoduchý způsob vytvoření umělé datové sady je takzvaná kompozice[2]. Jedná se o využití reálným obrázků a různého míchání s dalšími objekty s různým pozadím v 2D grafickém editoru, a případně využití dalších grafický úprav. Tento proces lze samozřejmě automatizovat a přidat parametry pro náhodnost. Problémem je získání samotných obrázků, především proto, že je zapotřebí objekty ideálně oddělit od jejich původního pozadí. Proto tento proces může být náročný z ohledu manuální práce. Případně lze vytvořit si své vlastní fotografie s jednobarevným pozadím, a tím si práci urychlit a ulehčit.

2.2.2 Generativní adversiální síťe

Vytváření syntetické datové sady pomocí GAN[8] (Generative Adversarial Network) je postup, který vygeneruje data, zhodnotí jak reálně vypadají, a dle toho se další vygenerovaná data vylepšují. Skládá se ze dvou hlavních částí, generátor a diskriminátor. Generátor vytváří náhodné data, z počátku buď dle náhodného šumu nebo začátečních dat. Tyto vygenerovaná data se pošlou do diskriminátoru. Diskriminátor poté rozhoduje, jak moc reálně nebo uměle vypadají data. Aby tohle diskriminátor mohl provést, tak potřebuje vědět, jak vypadají reálné data. Je tedy potřebné již mít vytvořenou reálnou datovou sadu, proto vygenerovaná data ve výsledku pouze vylepšují model, ale nelze vytvořit datovou sadu sama o sobě.

2.3 Automatizace vytváření datových sad

Můj skript jsem chtěl maximálně automatizovat. Výsledný skript lze spustit znalým uživatelem, a následně provede vše bez potřeby manuálního zásahu. V případě přerušení v prů-



Obrázek 2.8: Hlavní okno GUI programu. Lze si interaktivně vytvořit náhled, procházet vytvořenými soubory, začít proces vykreslování, či vytvořit konfigurační soubory pro datové sady.

běhu procesu lze skript spustit znovu a proces začne pokračovat u posledního vytvořeného snímku.

Dále jsem vytvořil program s uživatelským rozhraním, který lze vidět na obrázku 2.8. Ten jsem vytvořil za účelem pohodlnějšího hledání správných parametrů, o kterých budu psát v dalších kapitolách. Tento program má mimo jiné možnost vykreslit pouze jediný snímek jako náhled, a zobrazit ho ihned v programu, čímž se mi značně usnadnilo zobrazování výsledného snímku a to jak vypadá se změnami které jsem právě provedl. Hlavně při samotném vývoji skriptu byl velmi užitečný při vyhledávání různých chyb a nalezení vhodných minimálních a maximálních mezí parametrů. Díky možnosti vytvoření snímku pomocí počáteční hodnoty generátoru náhodných čísel (RNG seed) bylo jednoduché trasovat různé chyby. Každopádně po vytvoření konečné verze skriptu jsem program moc nevyužíval, poté jsem ho používal spíše jen pro interaktivní vytváření konfiguračních souborů pro jednotlivé datové sady.

2.4 Konvoluční neuronové sít

Konvuluční neuronová síť[7] (Convolutional Neuron Network, dále jen CNN) je typ umělé neuronové sítě, která se často používá pro analýzu obrazu. Její hlavní částí jsou právě konvoluční vrstvy které obsahují filtry, a mezi těmito vrstvami jsou obvykle takzvané pooling vrstvy. CNN jsou schopné se naučit rozpoznávat jednotlivé části obrazu jako jsou hrany nebo textury, které dokáží spojit a detekovat více komplexnější objekty. Tyto sítě mají mnoho aplikací, jako je například rozpoznávání obrazu, detekce objektů, nebo třeba klasifikace obrázků.

2.4.1 Vrstvy neuronové sítě

Konvoluční vrstvy

Konvoluční vrstvy se skládají z filtru, který se pohybuje po vstupních datech (obrázek), a po celé ploše aplikuje tenhle filtr. Výstupem je 3D nebo 4D tenzor[8] dle architektury, která tvoří prvek výstupní mapy příznaků (feature map). Tohle umožňuje detekci různých rysů, jako jsou textury, hrany a další charakteristiky objektu.

Každý filtr v konvoluční vrstvě má své váhy[9], které se během trénování učí, aby byly schopné zachytit relevantní vzory ve vstupních datech. Konvoluční vrstvy mohou být doplněny o takzvanou aktivační funkci jako je Sigmoid, SoftMax nebo ReLU (Rectified Linear Unit), která umožňuje zachytit více komplexnější vzory.

Konvoluční vrsta má 4 důležité parametry[3]:

- Velikost jádra (kernel size): Velikost prozkoumávané oblast. Menší rozměry lépe zachycují jemnější detaily a rysy, zatímco větší jádra lépe zachytí větší vzory. Nejčastěji se používá velikost 3×3 nebo 5×5.
- Krok (stride): O jaký krok se jádro posouvá po vstupních datech. Větší hodnota je výpočetně rychlejší a může pomoci proti přeučení (overfitting).
- Rozměr vstupní matice: Rozměr obrázku, včetně hloubky, která v tomhle kontextu značí barvu. Při barevných obrázcích se pracuje na RGB kanálech jednotlivě.
- Nulové doplnění (zero padding): Metoda pro rozšíření vstupních dat, která zachová rozměr vstupní dat pro efektivnější extrakci vzorů. Také umožňuje lépe zachytit data z okrajů vstupních dat a zlepšit tím výkon zejména u CNN architektur s více vrstvami.

Pooling vrstvy

Pooling vrstvy zmenšují rozměr prostoru příznaků pomocí takzvané pooling operace na výstupy předchozích vrstev.

Pooling operace (příklad v obrázku 2.9) je agregace informací v rámci malých pooling oblastí a vytváří z nich jednoduchou reprezentaci. Běžnými typy pooling operace je max pooling a average pooling. V max pooling se zachovávají pouze nejvýznamnější rysy v dané oblasti, zatímco v average pooling je vypočten průměr všech prvků.

Pooling vrstvy snižují výpočetní náročnost, potlačují přeučení (overfitting), a zvyšují robustnost díky tomu, že přesunutí objektu v obraze má minimální vliv na výstup sítě.

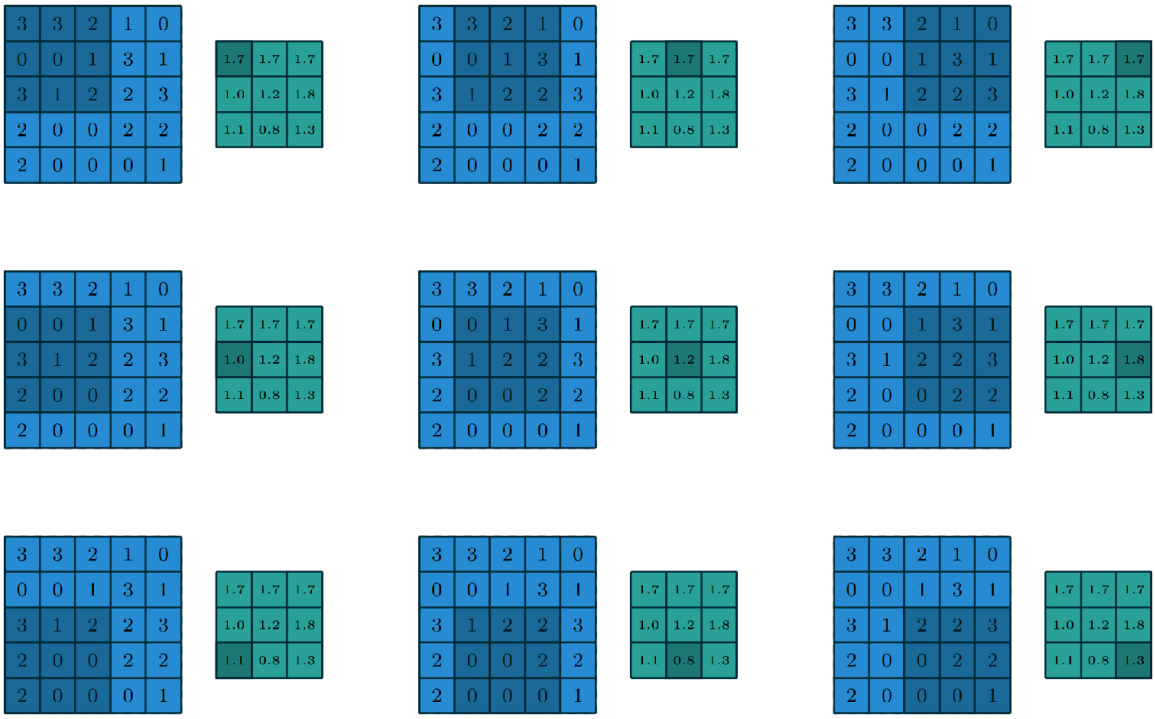
Plně propojené vrstvy

V plně propojených vrstvách jsou všechny neurony propojeny s každým neuronem v předešlé a následující vrstvě. Každý signál vstupního neuronu má specifickou váhu, která určuje, jak moc je důležitý pro výstup neuronu. Každý neuron ve vrstvě může mít přidán bias[10]. Po

[8] Tenzor je základní datová struktura používaná pro reprezentaci vstupních dat, výstupů sítě, vah a gradientů během trénování.

[9] Váhy jsou parametry spojení mezi neurony. Každý vstupní signál je vážen určitou vahou, která určuje, jak moc je důležitý pro výstup neuronu. Váhy se mění během trénování tak, aby se model naučil reprezentovat data a provádět požadované úkoly.

[10] Bias je konstantní hodnota, která se přidává k vážené sumě vstupů neuronu před aplikací aktivační funkce. Pomáhá modelu přizpůsobit se vstupům posunutím aktivační funkce nahoru nebo dolů.



Obrázek 2.9: Výpočet výstupních hodnot 3×3 s average pooling na vstupu 5×5 pomocí kroků (stride) 1×1. Převzato z [5].

vážené sumě může být na každém neuronu aplikována aktivační funkce, která přidává síti schopnost modelovat komplexní nelineární vstupy mezi vstupy a výstupy.

Plně propojené vrstvy jsou většinou používány v konečných částech neuronové sítě, například pro klasifikaci, kde provádějí rozhodování na základě extrahovaných příznaků z předchozích vrstev.

2.4.2 Aktivační funkce

Aktivační funkce slouží pro přidání nelinearity do modelu, aplikují se na výstupy neuronů na jednotlivých vrstvách. Bez aplikační funkce by měl model problém modelovat více komplexnější vztahy mezi daty.

Nejznámější aktivační funkce jsou:

- ReLU (Rectified Linear Unit): Vrací 0 pro všechny záporné vstupy a výstupy.
- Sigmoid: Převádí vstupy na hodnoty mezi 0 a 1, což je užitečné pro binární klasifikace nebo pro výstupy, které mají být interpretovány jako pravděpodobnost.
- Hyperbolický tangens (TanH): Podobná jako Sigmoid, ale převádí hodnoty do rozmezí -1 až +1.
- Softmax: Převádí výstupy na pravděpodobnostní rozložení, kde každá hodnota představuje pravděpodobnost určité třídy.

2.4.3 Ztrátová funkce

Ztrátová funkce (loss function) je matematická funkce, která vyhodnocuje rozdíl očekávaným výstupem sítě a opravdovým výstupem. Tedy poskytuje zpětnou vazbu, která řídí proces optimalizace parametrů modelu. Jedná se o metriku, která určuje, jak dobře se model učí predikovat výstupy na základě vstupních dat během trénování.

Je definována jako funkce parametrů vah a bias, která bere vstupní data a odpovídající cílové hodnoty a vrací hodnotu ztráty, která indikuje, jak dobře se model chová na těchto datech. Cílem trénování je tuhle hodnotu minimalizovat.

Existuje několik ztrátových funkcí, které se více hodí na jiné práce:

- Kategorická křížová entropie (Categorical Cross-Entropy): Pro klasifikační úlohy s více třídami. Funkce porovnává pravděpodobnosti přiřazené modelem k jednotlivých třídám.
- Binární křížová entropie (Binary Cross-Entropy): Obdobná jako kategorická křížová entropie, ale používá se pro binární klasifikaci. Výstupní prostor je omezen na dvě třídy.
- Mean Squared Error (MSE): Pro regresní úlohy. Model predikuje kontinuální hodnoty. Funkce porovnává skutečné a predikované hodnoty.

2.4.4 Gradientový sestup

Gradientový sestup (gradient descent) je optimalizační algoritmus používaný k nalezení minima nebo maxima funkce cílové ztrátové funkce. V CNN se používá k nalezení optimálních parametrů modelu, které minimalizují hodnotu ztrátové funkce.

Postup gradientového sestupu zahrnuje následující kroky:

- 1. Inicializace parametrů modelu, buď náhodně nebo pomocí heuristik.
- 2. Výpočet gradientu: Pro každou iteraci algoritmu se vypočítá gradient ztrátové funkce vzhledem k parametrům modelu.
- 3. Aktualizace parametrů: Parametry modelu jsou aktualizovány v souladu s gradientem a rychlostí učení.
- 4. Opakování: Kroky 2 a 3 se opakují, dokud není splněna některá ukončovací podmínka, například dosažení stanoveného počtu iterací.

Stochastický gradientový sestup (Stochastic Gradient Descent, SGD) je konkrétní variantou gradientového sestupu, která se používá především při trénování velkých datových sad. Tento algoritmus má nízké paměťové nároky, protože aktualizuje parametry modelu postupně pro každou dávku dat. Existují také další varianty gradientového sestupu, jako je například mini-batch gradient descent nebo momentum SGD, které přinášejí další optimalizační techniky pro rychlejší a stabilnější trénování.

2.4.5 Zpětná propagace

Zpětná propagace (back propagation) je způsob, kterým se vypočítají gradienty ztrátové funkce vzhledem k parametrům modelu, a tím umožnuje efektivně aktualizovat tyhle parametry pomocí optimalizačního algoritmu, třeba gradientovým sestupem.

Princip se skládá ze 4 kroků:

- 1. Předpředání (forward pass): Provede se předání dat od vstupních až do výstupních vrstev. Během toho se vstupní data transformují skrz vrstvy sítě pomocí váh a aktivačních funkcí až do výstupní vrstvy, kde se vypočítá predikce modelu.
- 2. Výpočet ztráty: Vypočítá se ztrátová funkce, která vyhodnotí rozdíl mezi skutečnými a predikovanými hodnotami.
- 3. Zpětná propagace: Vypočte se gradient ztrátové funkce vzhledem k parametrům modelu. Proces se provádí rekurzivně. Nejprve se vypočítají gradienty na výstupní vrstvě a pak se postupně propagují až ke vstupní vrstvě. Během toho se gradienty postupně násobí s váhami jednotlivých spojení a aktivačními funkcemi pro každou vrstvu.
- 4. Aktualizace parametrů: Pomocí vypočítaných gradientů se aktualizují parametry tak, aby minimalizoval ztrátovou funkci.

2.4.6 Trénování CNN

Trénování má několik parametrů – v kontextu CNN se tyto specifické parametry nazývají hyperparametry. Nastavují se před začátkem trénování a mají značný vliv na úspěšnost modelu, proto je důležité s těmito hyperparametry experimentovat a nalézt ty nejlepší hodnoty.

Jedná se o hyperparametry jako je například rychlost učení (learning rate), velikost dávky (batch size), počet epoch, velikost vrstev, aktivační funkce, L1/L2 penalizace nebo vynechání neuronů (dropout).

Rychlost učení je kritický hyperparametr pro učení. Velká hodnota může vést k neschopnosti konvergovat a nebo k oscilaci kolem minima ztrátové funkce. Naopak příliš malá hodnota způsobí velmi pomalé trénování, a také může způsobit, že síť zůstane uvězněna v lokálním minimu. Proto je důležité nejen zvolit správnou hodnotu, ale taky správný mechanismum na postupné snižování rychlosti učení.

2.5 Trénování mého modelu a post-processing

Pro samotnou detekci využívám neuronovou síť CenterNet[15]. CenterNet je založený na takzvaných *center points*, neboli bodové reprezentaci objektů, základní myšlenka je tedy v identifikaci středu objektu. Tato neuronová síť může využívat architekturu Hourglass-104[10], ale lze využívat i jiné. Já jsem ve většině případů používal výchozí architekturu DLA-34[14] (Deep Layer Aggregation), která je jednodušší a méně výpočetně náročná než právě Hourglass. Pro mé učely by samozřejmě vyhovovali i jiné, jednodušší neuronové sítě, každopádně jsem skončil u CenterNetu.

Pro vyřešení různých problému a pro vylepšení přesnosti, provádím po samotné detekci další zpracování. Zde se shlukují detekce pro vyfiltrování nepravdivých pozitiv, nebo se domýšlejí predikce tam, kde se nenalezly, ale logicky by tam pravděpodobně měly být.

Dále je problém se značným odrazem u fólií, jak lze vidět na obrázku 2.10. Tohle lze vyřešit tím, že se nepočítají přepravky z jediného snímku, ale využije se fakt, že vstupem je videozáznam, a tedy snímky ve kterých se detekuje podprůměrný počet počet přepravek se vyfiltrují, a tím se lépe přiblíží k opravdovému počtu přepravek.

Obrázek 2.10: Sled snímků z videozáznamu, který vyobrazuje odrazy na fólii dle pozice přepravek. V prvním snímku lze jednoduše rozpoznat pravý sloupec s přepravkami, v snímku 2 je v tohle sloupci silný odraz, který téměř znemožňuje spolehlivě detekovat 5 až 8 přepravek. V dalším snímku se tohle opakuje u levého sloupce. Nakonec u posledních dvou snímků není prominentní odraz, a bylo by tedy značně jednodušší zde spolehlivě detekovat přepravky. Snímky byly oříznuty jen na podstatné části obrázku pro lepší čitelnost.

Kapitola 3

Návrh

3.1 Skript na vytvoření syntetických dat

V obrázku 3.1 je obecný návrh postupu vytvoření datových sad. Jako první je potřeba udělat alespoň dvě datové sady z reálných dat, jednu která se použije pro testování, a druhou ze které se neuronová síť bude učit v případech, kdy se bude určovat vztah mezi reálnými a syntetickými daty. Jedná se tedy hlavně pro výsledné porovnání úspěšnosti u detekce ve statických snímcích a ve videozáznamu.

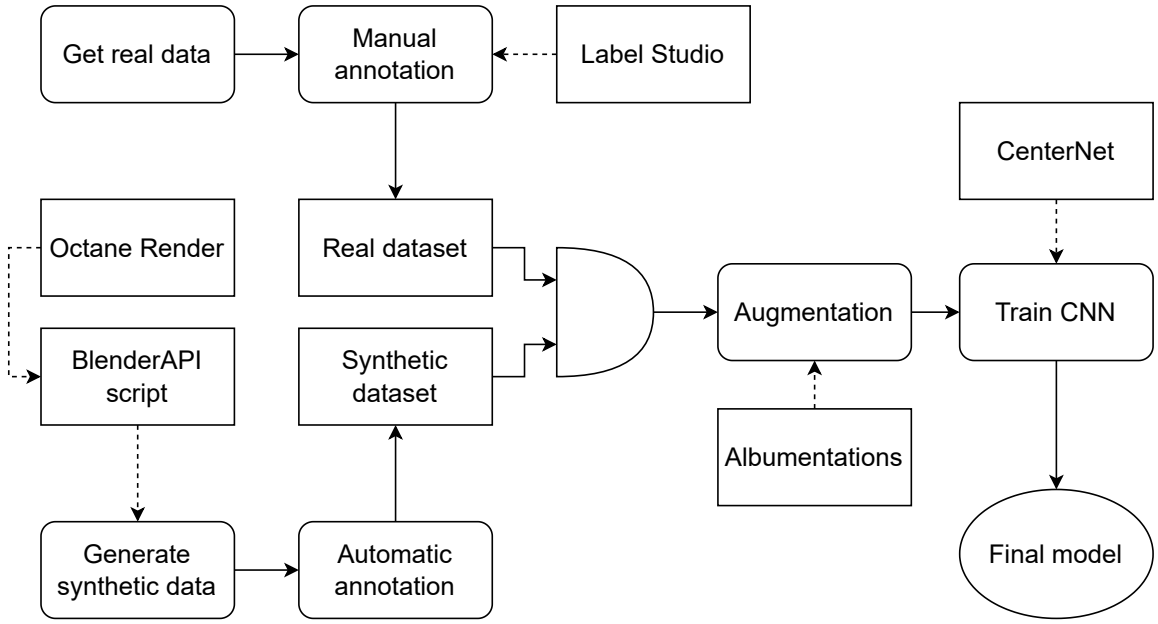
Reálné datové sady byly anotovány pomocí programu Label Studio[13]. U testovací sady jsem začernil ty objekty, které byly na okraji snímku a byly tedy značně oříznuté, jelikož by poté bylo obtížně rozhodovat o tom jestli je rozumné očekávat, aby je detektor predikoval. Dále jsem vytvořil dva videozáznamy pro testy ve videu, jeden s přepravkami E1, a druhý s přepravkami E2.

Pro vytvoření syntetických dat se použil program Blender s Octane Render jako vykreslovací engine. Skript nakonec vytvoří kompletní datovou sadu, není potřeba ji nijak rozdělovat na trénovací a validační snímky (případně i testovací snímky, každopádně si myslím, že v tomhle případě je více objektivní jako testovací sadu vždy použít reálné data), tohle se nastavilo před spuštěním skriptu do konfiguračního souboru, který ukládá všechny informace o dané datové sadě. Výstup skriptu je kompletní datová sada připravená pro trénování v síti CenterNet. Pro manipulaci s datovými sadami, jako je například kombinace syntetický a reálných dat, jsem vytvořil skripty zvlášť.

3.2 Postup trénování

Při trénování se použije augmentace dat, ale to pouze u pár posledních experimentů. U většiny mých experimentů nebylo smyslem vytvořit skvělý model s vysokou úspěšností, ale většinou se jednalo o to, aby jsem zjistil relativní změny v porovnání s ostatními modely, tedy jaké byly změny, když jsem něco změnil při vytváření datové sady.

Augmentace by proto představovala zbytečnou komplikaci, nebo dokonce by mohla způsobit výrazně odlišně rozdíly při stejných podmínkách. Každopádně u posledních modelů jsem pár jednoduchých augmentací použil, a to za pomocí knihovny Albumentations[4]. Albumentations je velmi populární a často používaná knihovna pro augmentace, a jelikož jsem neměl na augmentaci nějaké speciální požadavky, tak jsem také neměl důvod hledat něco alternativního.



Obrázek 3.1: Architektura vytvoření modelu. Dle účelu se používaly jenom syntetické data, nebo jenom reálné, a nebo kombinace obou typů (pokusy s kombinacemi obou typů nebyly moc úspěšné, takže jsem je nakonec vynechal). Augmentace jsem často nevyužíval, aby jsem viděl přesnější výsledky, bez spoléhání na to, že augmentace zacelí možné nedostatky.

3.2.1 Způsob testování a testovací sady

Měl jsem dvě testovací sady s reálnými snímky. První sada je třídně nevyvážená sada s dohromady 2525 vzorky, tuhle sadu jsem používal pouze při vyhodnocování reálných nebo hybridních datových sad. Druhá sada je třídně vyvážená, obsahuje 1000 vzorků přepravky E1, a dalších 1000 vzorků přepravky E2. Tuhle testovací sadu jsem použil u všech plně syntetických datových sad. V reálných či hybridních datových sadách jsem použil snímky z této sady, a pro testování jsem poté využil 1. testovací sadu. I přesto, že je tahle sada třídně nevyvážená, tak si nemyslím že se jedná o problém, protože sada má méně přepravek E2, zatímco dle výsledků z experimentů je obtížnější model natrénovat na přepravku E1, zatímco E2 je bezproblémová.

Používal jsem běžné metriky pro vyhodnocování úspěšnosti datových sad. Testovací skript vypočítal průnik přes sjednocení (Intersection over Union) pro každou predikci od detektoru a porovnal ho s referenčními pravdami testovací sady. Vyhodnocoval jsem na hladině $IoU = 0.4$. Ve většině případů se doporučují hodnoty používají hodnoty 0.4 až 0.6, a jelikož jsem si nebyl moc jistý mými manuálně anotovanými testovacími daty, tak jsem zvolil více tolerantní hodnotu, jelikož mé manuální anotace byly menší než ty vytvořené skriptem. Pokud je tato plocha průniku přes sjednocení větší než nastavený práh, a zároveň se třída predikce a referenční pravdě shoduje, tak se predikce vyhodnotí jako pravdivý pozitiv (true positive). Dále skript vyhodnocuje pravdivý negativ (true negative), nepravdivý pozitiv (false pozitiv) a nepravdivý negativ (false negative). Tyto elementy skládají takzvanou matici záměn[6] (confusion matrix). Každopádně pravdivý negativ, který by indikoval správně identifikovanou nepřítomnost objektu, je v kontextu detekce objektů obvykle zanedbáván. Detekce objektů se primárně zaměřuje na identifikaci přítomnosti objektů, nikoli na jejich nepřítomnost. Proto tuhle metriku nechci využívat, jelikož metrika accuracy (která

dává v potaz všechny elementy z matice záměn, včetně právě pravdivý negativ), by byla ovlivněna nepřítomností objektů. Vyhodnocovat nepřítomnost se mi zdá zavádějící, jelikož detektor CenterNet vrací 100 predikcí s největší úrovní jistoty (confidence score), zatímco ve scéně se vyskytuje většinou 10-34 přepravek, proto velké množství predikcí je pouze pseudonáhodný šum pozadí, který nic zajímavého neznačí.

Z hodnot matice záměn lze dále vypočítat pro mě více podstatné metriky. Konkrétně při testech parametrů scény jsem se řídil metrikou F1-skóre, která je kombinací *precision* (PPV – Positive Predictive Value, občas nazýváno *confidence*) a recall (TPR - True Positive Rate, občas nazýváno *sensitivity*).

Vzorce pro následující metriky jsou následující:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

$$precison = \frac{TP}{TP + FP} \tag{3.2}$$

$$recall = \frac{TP}{TP + FN} \tag{3.3}$$

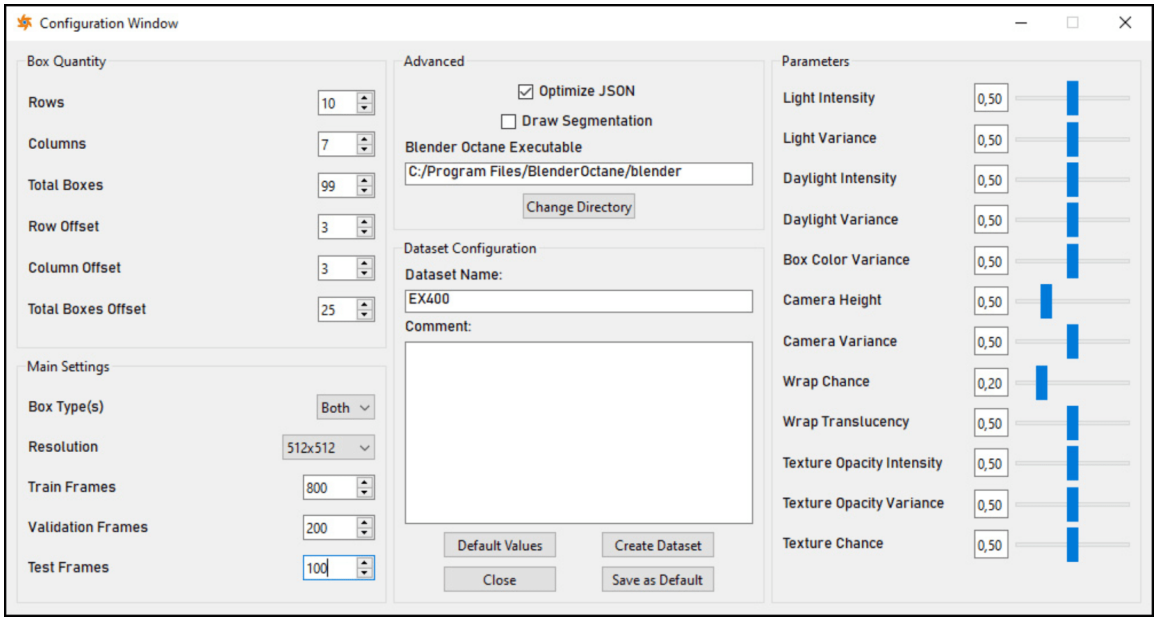
$$F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.4}$$

3.3 Volba správných hodnot parametrů scény, jejich mezí a samotný postup

Výsledný model je jenom jeden, proto je důležité použít parametry scény, které natrénují model na co největší úspěšnost, a s co nejméně nesprávných detekcí. Pro nalezení té správné kombinace hodnot parametrů bylo potřeba vytvořit stovky datových sad a porovnávat mezi nimi výsledky. Každá datová sada mění pouze jeden parametr, jinak by nebylo možné identifikovat, proč se změnila úspěšnost. Je ale zapotřebí zvolit vhodné počáteční parametry a hlavně rozmezí těchto hodnot, a zde nastává možný problém. Tato metoda nelze dobře zautomatizovat – nebo lépe řečeno, bylo by to možné, ale to by potřebovalo vytvořit stovky, ne-li tisíce datových sad, což by bylo výpočetně a časově náročné. Proto je potřeba zvolit hodnoty počátečních parametrů takové, které mají nejmenší šanci zneplatnit modely, a zároveň omezit minimální a maximální hodnoty. Proto jsem zvolil, že vytvořím takové parametry, které se budou pohybovat v normalizovaném rozsahu 0.0 až 1.0, s tím že v první fázi budu krokovat s hodnotou 0.2. O tom, co přesně jednotlivé parametry ovlivňují, píši v kapitole 5.2.2. Proces hledání ideálních parametrů je rozdělen do 4 fází. Pro jednodušší vytváření datových sad jsem si vytvořil nové okno do mého programu s rozhraním, který mi tohle usnadní. Tohle nové okno lze vidět na obrázku 3.2. Při vytvoření datové sady se vytvoří konfigurační JSON soubor, který obsahuje všechny potřebné informace, jako jsou použité parametry a jaké snímky už byly vykresleny.

První fáze má za úkol otestovat všechny parametry s výchozími hodnotami, kromě současně testovaného parametru, jedná se tedy o velmi obecné rozřazení. Existuje set experimentů pro každý parametr, kde je každý experiment identický, kromě hodnoty testovaného parametru. Krokuje se s hodnotou 0.2, tedy každý set bude mít experimenty s výchozími parametry a testovacím parametrem s hodnotami 0.0, 0.2, 0.4 atd. Jedná se tedy o celkem 77 experimentů (parametr výšky kamery je v mezích 0 až 2).

Druhá fáze zkombinuje ty nejúspěšnější hodnoty parametrů z předešlé fáze. U každého parametru se vybere ta hodnota, která měla nejvyšší úspěšnost. Tyto hodnoty budou



Obrázek 3.2: Konfigurační okno programu. Lze si zde interaktivně zvolit parametry pro vytvoření datové sady, jako je počet a rozdělení snímků, které typy přepravky se budou vytvářet, parametry scény, a nebo třeba vytvoření dodatečných obrázků, které mají v sobě nakreslené anotace.

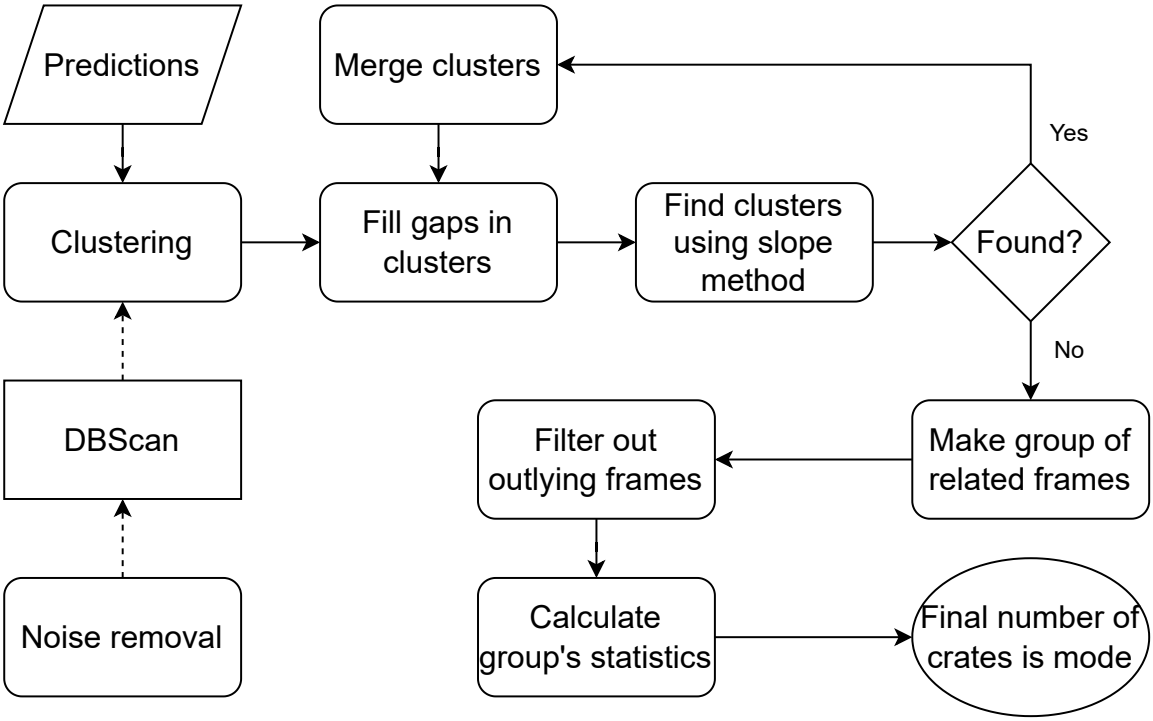
fungovat jako nové výchozí hodnoty. V případě, že jiná hodnota má obdobnou úspěšnost, tak se tato hodnota také otestuje, jelikož je možné, že parametr bude mít více lokálních maxim. Například při menších hodnotách parametrů osvětlení může být model více úspěšnější pro detekci v horších světelných podmínkách, zatímco s větší intenzitou světla bude více úspěšnější pro detekci v dobrých světelných podmínkách. Ani jedna z těchto možností není zaručeně ta správná, proto je zapotřebí prozkoumat obě. Tedy vytvoří se datové sady, které budou využívat ty nejúspěšnější hodnoty každého parametru, a obdobně jako u první fáze, se budou testovat jiné hodnoty parametrů, ale teď se budou testovat pouze ty hodnoty, které se blížily těm nejúspěšnějším. Je tedy šance, že konkrétní parametr může mít pouze 1 hodnotu s vysokou úspěšností, zatímco jiné parametry jich mohou mít několik.

Třetí fáze je o detailnějším nalezení ideálních hodnot, tedy jedná se o nejlepšího kandidáta z druhé fáze, a budu testovat sousední hodnoty parametrů s jemnější krokováním, tedy místo krokování hodnotou 0.2, budu krokovat o 0.1. Například když nejlepší hodnota parametru bude 0.6, tak v tomhle kroku se otestují hodnoty 0.5 a 0.7.

Čtvrtá fáze je obdobou třetí fáze, ale s ještě mírnějším krokováním 0.05, tedy $\frac{1}{4}$ původního kroku. Pokud nejlepší výsledek v třetí fázi byl 0.5, tak čtvrtá fáze otestuje sousední hodnoty 0.45 a 0.55.

3.4 Post-processing

Obrázek 3.3 popisuje postup zpracování detekce. Probíhá po vytvoření predikcí na bázi jednotlivých snímků, a následně z těchto informací různě filtruje a domýšlí možné predikce, jedná se tedy o následné zpracování (post-processing). Jako první se použije shlukovací algoritmus DBScan. Tento algoritmus jsem neimplementoval sám, ale využil jsem knihovnu



Obrázek 3.3: Postup detekce s využitím post processing. Vstupem jsou predikce z detektoru, které se shlukují pomocí algoritmu DBScan, a následně se domýšlejí predikce tam, kde by se logicky měly nacházet. Následně se vytvoří skupina takto upravených snímků, odfiltrují se podezřelé snímky, a z jejich statistiky se vezme počet přepravek.

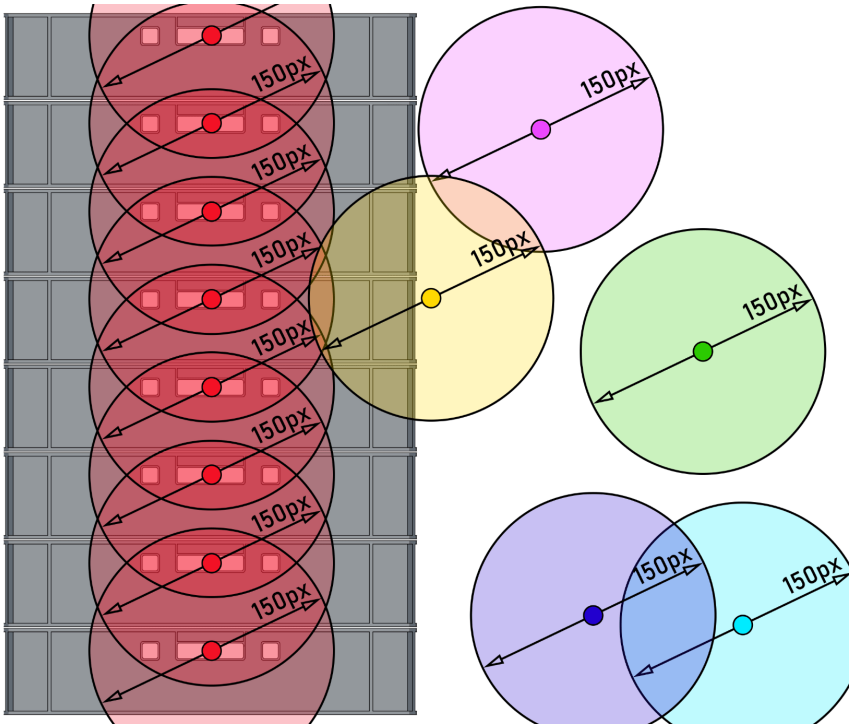
Scikit-learn[11], tato knihovna má také skvělou dokumentaci[1] s popisem ostatních shlukovacích algoritmů, která mi pomohla vybrat ten správný algoritmus pro moji práci.

Shlukovací algoritmus DBScan shlukuje nejbližší body (v tomhle kontextu středy predikce od detektoru). Uživatel si může nastavit parametry na vzdálenost mezi těmito body, tahle vzdálenost se jmenuje ϵ. V případě, že je bod osamostatněn a je mezi tímto bodem a ostatními velká mezera, tak se nemůže s jiným bodem sloučit. Tedy nebude částí žádného shluku, proto je považován za šum, a tím se tedy vyfiltruje, je totiž předpoklad, že se jedná o nepravdivý pozitiv. Následně je pokus o domyšlení chybějících predikcí u každého shluku.

Na obrázku 3.4 každý bod představuje predikci detektoru, zahrnující směs pravdivých pozitiv a nepravdivých pozitiv. Cílem algoritmu DBScan je identifikovat a vyfiltrovat pouze nepravdivé pozitivy. Správné detekce jsou zvýrazněny červenou barvou v levé části obrázku, které odpovídají přepravkám v pozadí. Parametr ϵ určuje maximální vzdálenost, ve které každý bod hledá své okolní sousedy, v tomhle případě je nastaven na 150 pixelů. Červené body úspěšně vytvořily jeden shluk, protože vzdálenosti mezi nejbližšími body jsou vždy menší než 150 pixelů. Body, které nemají v dosahu žádného souseda v rámci ϵ, nejsou součástí žádného shluku, protože minimální počet bodů pro vytvoření shluku je stanoven na $n = 2$. Tyto body jsou tedy úspěšně vyfiltrovány.

Výsledné shluky jsou také užitečné pro domyšlení detekcí v místech, kde není žádná předpověď, jak je zobrazeno v obrázku 3.5. Při identifikaci přepravek ležících na sobě je logicky nepravděpodobné, že by mezi nimi vznikla mezera. Pokud je však hodnota parame-

[1] Scikit-learn shlukovací algoritmy: scikit-learn.org/stable/modules/clustering



Obrázek 3.4: Vizualizace filtrování nepravdivých pozitiv pomocí shlukovacího algoritmu DBScan. Každá barva reprezentuje jeden shluk. Pokud se do shluku nespojily alespoň dva body, tak se bod odfiltruje.

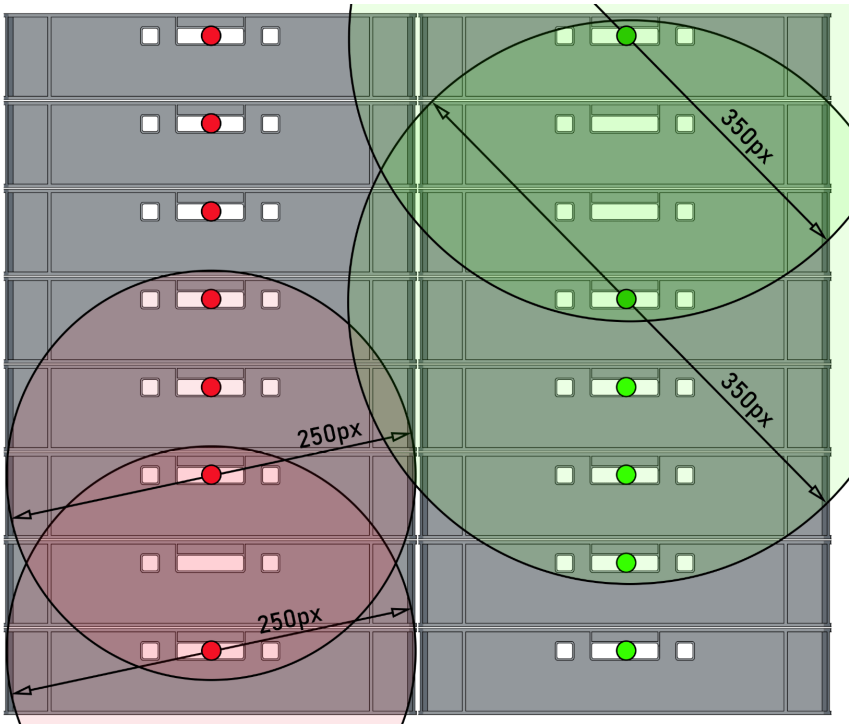
tru ϵ dostatečně tolerantní, aby shlukovala body i v případě, že mezi nimi chybí predikce, vytvoří se shluk s mezerou. Tuto mezeru lze detekovat a lze tedy doplnit predikci v těchto oblastech.

Lze také vidět, že se v levém sloupci nepodařilo detektoru správně predikovat druhou přepravku od zdola. Každopádně hodnota ϵ je dostatečně tolerantní, že predikce z přepravky 1 a 3 se sloučily do stejného shluku. Následně lze vypočítat průměrné mezery mezi body, a v případě, že se nalezne mezera, která je značně nad průměrem, tak lze předpokládat, že se zde má nacházet další přepravka. Domyslí se zde tedy další predikce.

V pravém sloupci lze vidět, že se nepodařilo detekovat přepravky 2 a 3 z vrchu. Každopádně ϵ je velice tolerantní, a přepravky 1 a 4 se navzájem sloučili, a vznikl tedy shluk s velkou mezerou. Zároveň jelikož jsou přepravky nízké a zároveň široké, tak ani značná velikost ϵ nezasahuje do vedlejšího sloupce. Obdobně jako v prvním případě se zde domyslí predikce, ale jelikož mezera mezi přepravkami 1 a 4 je zhruba 3× větší než průměr, tak se zde domyslí 2 predikce.

Existuje další metoda, která může detekovat chybějící přepravky. Lze ji vidět na obrázku 3.6. Pro každý shluk se vypočítá jeho směr a střed. Následně se zadá šířka pro prozkoumání a tolerance úhlu. Pokud například shluk A má svůj směr 5°, shluk B 6°a pokud tolerance úhlu tyhle dvě hodnoty povoluje, tak se tyto dva shluky otestují, jestli se jejich cesty překříží. Jestli se plocha shluku A dotkne středu shluku B, tak se z nich vytvoří jeden shluk. Následně se ve shluku nalezne mezera a domyslí se chybějící body, stejně jako v předešlé metodě.

Na obrázku 3.7 je vidět rozhraní detektoru. Tento detektor slouží jako náhled pro detekce na statickém snímku, lze si zde u více obrázků vyzkoušet, jaké kombinace parametrů budou



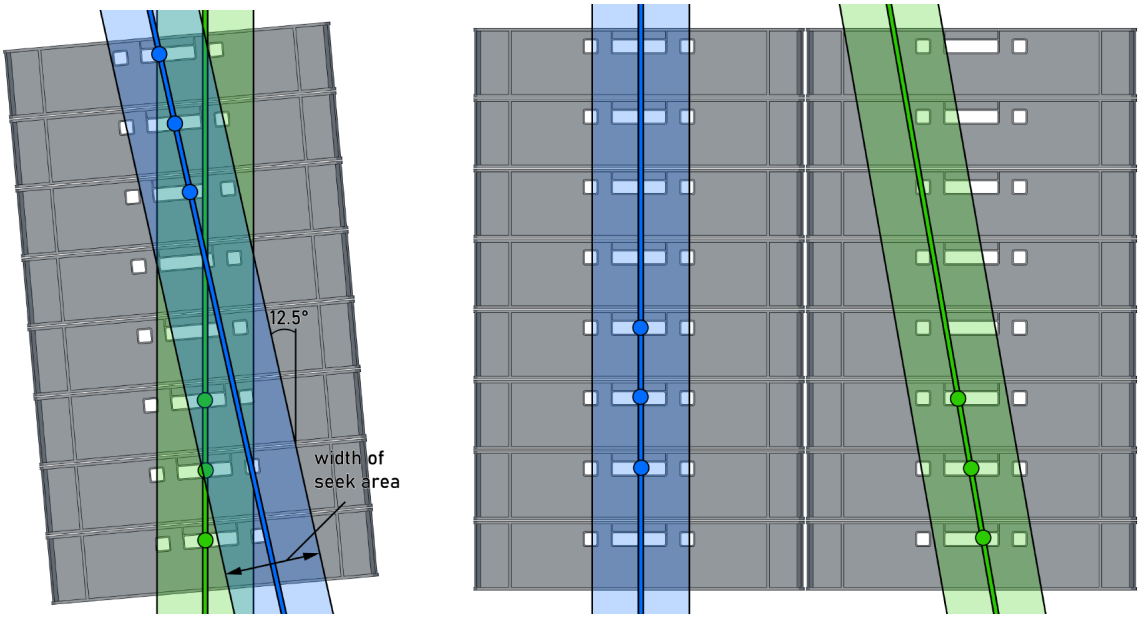
Obrázek 3.5: Vizualizace shlukování. Body jsou predikce z detektoru, stejné barvy značí, že body náleží do stejného shluku. Jsou zde dvě hodnoty ϵ, 250 pixelů a 350 pixelů. Jedná se pouze o vizualizaci více věcí v jednom obrázku, v praxi se používá jen jedna hodnota ϵ. Pro přehlednost je zobrazen dosah ϵ pouze pro 4 relevantní body.

mít nejlepší výsledky, a tím si zvolit tu správnou kombinaci, nebo v případě, že post-processing nebude vhodný, tak ho lze vypnout. Lze si nastavit parametr práh detekce, a dále jsou k dispozici další 4 parametry, které ovlivňují tři dříve zmíněné metody na domyšlení predikcí a filtrování nesprávných predikcí. Program také vypíše všechny predikce a statistiky, tedy kolik jich predikoval detektor, kolik se jich domyslelo, kolik se odfiltrovalo predikcí, a kolik se vytvořilo shluků. Program vizualizuje body ve stejném shluku pomocí stejné barvy a také je čísluje, to je ale spíše pro ladění algoritmů.

Dále je na obrázku 3.8 porovnání snímku před a po využití metod následného zpracování. Všimněte si na snímku, že v levém sloupci chybí jedna detekce mezi body 19 a 10. Tahle detekce se domyslela při použití metody následného zpracování, když se z celého levého sloupce stal jeden shluk díky dostatečné toleranci ϵ u DBScan algorithmu, a poté co se vytvořil shluk, tak se nalezla tato mezera a nakonec se doplnila.

Poté v pravém sloupci je vidět, že detektor měl velké problémy s detekcí přepravek v pravém sloupci, když jsou přepravky zakryté fólií. Zde zafungovala poslední metoda, tedy průnik shluků. Dva spodní body a čtyři horní body spolu vytvořily dva shluky, a jelikož oba směřovali stejně (nebo alespoň dostatečně stejně, aby byly v toleranci), tak se potkali a vytvořili spolu jeden shluk. Následně se v mezeře domyslelo 7 bodů, i když ve skutečnosti je zde 8 přepravek, každopádně i přesto se výrazně zvýšila v tomhle konkrétním případě úspěšnost.

Z původních 22 správných a 0 nesprávných predikcí (obrázek neukazuje celý snímek, každopádně v obou případech se všechny potencionální nepravdivé pozitiva odfiltrovaly pomocí prahu jistoty predikce), s využitým následného zpracování se nalezla jedna predikce



(a) Vizualizace průniku shluků. Modrá a zelená barva znázorňují elementy patřící ke svému shluku, body jsou predikce, hrubá čára je směr shluku, a průhledná vrstva je plocha tolerance průniku. Jelikož se překrývají se středem druhého shluku, tak se sloučí dohromady. Efekt byl zdůrazněn pro lepší vizualizaci

(b) Vizualizace průniku shluků, kdy k průniku nedochází. Směr levého shluku je o 10° rozdílný od modrého shluku. I kdyby byla tolerance směru shluků velká a tyto dva shluky by se otestovaly na průnik, metoda by je nespojila dohromady v jeden shluk.

Obrázek 3.6: Vizualizace průniku shluků.

pomocí metody shlukování, a dalších sedm predikcí se nalezlo pomocí metody průniku shluků. Tedy z úspěšnosti 64.7 % se pomocí post-processing zvýšila na 88.2 %. Každopádně se stále jedná pouze o detekci na statickém snímku, při detekci na videozáznamu by se tyto problematické snímky s odrazem fólií měly ideálně vyfiltrovat a být ještě více přesnější. Samozřejmě se jedná o více extrémnější příklad, kdy jsem využil horší model, aby šel lépe vidět efekt těchto metod.

3.5 Zpracování videa

Poslední částí je zpracování videa, je zde tedy šance využít více snímku k nalezení té správné hodnoty přepravek. V této práci se ale snažím prozkoumat více možností, takže není potřeba o výstupu tohoto videozáznamu uvažovat jako o finálním výsledku, v poslední kapitole se zabývám výsledky všech prozkoumaných možností – predikce pomocí detektoru, predikce pomocí detektoru a za použití post-processing algoritmů, predikce v obraze pomocí detektoru a algoritmů pro vyfiltrování nevyžádaných snímků, a nakonec predikce v obraze pomocí detektoru, algoritmů pro vyfiltrování nevyžádaných snímku a post-processing algoritmu na individuálních snímcích z videozáznamu.

Obrázek 3.7: Rozhraní programu na interaktivní detekci a nastavení následného zpracování. Lze si zde zvolit kombinaci parametrů, která uživatelovi vyhovuje. Je k dispozici 8 snímků, mezi kterými si lze prohlédnout, jak se program chová u jiných podmínek.

Hlavní princip vyfiltrování nevyžádaných snímků spočívá ve vytvoření správné skupiny snímků, které spolu souvisí, tedy aby se převoz jedné palety s přepravkami opravdu detekoval jako jeden převoz a jedna paleta. Tohoto jsem dosáhl pomocí detekce směru pohybu. Vypočítal jsem pro každý snímek střed predikcí, a porovnávám ho se středem předešlého snímku. Pokud je směr buď zleva doprava v dosahu 90°, tak se paleta s přepravkami přepravuje správně (či naopak, program lze nastavit pro přepravu ze skladu do dodávky nebo naopak), a informace o jednotlivých snímcích se začnou ukládat. Když se detekuje změna směru, tak se z počátku dá šance na zotavení, ale pokud se nezotaví, tak se usoudí že se opravdu změnil směr, a přestanou se ukládat snímky, a tím se nalezly všechny snímku ve skupině.

Když se vytvoří skupina souvisejících snímků, tak se ještě ze skupiny vytřídí některé snímky. Vždy to jsou některé počáteční a poslední snímky, jelikož když se přepravky dostanou do záběru kamery, tak se nikdy neobjeví všechny zároveň. Dále se vypočítá průměr ze skupiny, a dle tolerance se vytřídí ty snímky, které se až moc liší od ostatních. Tohle je většinou způsobené právě fóliemi, které pod daným úhlem odráží světlo přímo do kamery, a tím se značně zhoršují predikce od detektoru.

Ze zbývajících snímku se následně vypočítají její statistiky. Zjistil jsem, že hodnota modus nejpřesněji odhadne počet přepravek, které se ve videu převážely. Tohle číslo je tedy konečným počtem přepravek.

Obrázek 3.8: Příklad detekce. Levý obrázek je bez využití metod post-processing, pravý obrázek je s jeho využitím. Bez využití se detekovalo 22 z 34 přepravek, s využitím 30 z 34. V mezeře v levém sloupci se domyslela jedna predikce (číslo 7 v pravém obrázku), v pravém sloupci se domyslelo v mezeře dalších 7 predikcí (v této mezeře je ve skutečnosti 8 nedetekovaných přepravek).

Kapitola 4

Přínos práce pro odvětví syntetických datových sad

4.1 Výhody a uplatnění

Co se týče obecně vytváření syntetických dat pomocí CGI, největší uplatnění má tato metoda v odvětvích či konkrétních případech, kdy tyto data lze jenom obtížně získat, či je zapotřebí získat velké množství dat.

To mohou být případy, kdy výrobek má vytvořený koncept, ale ještě se nezačala výroba nebo se nezačal prodávat, nebo když se jedná o vzácný objekt, jako jsou třeba málo prozkoumaní živočichové, a nebo jednoduše objekty, pro které je obtížné najít reálné data, jak se například píše v článku[1] o problémech s nalezením dat pro specializované technické odvětví.

Tato metoda může potlačit nebo případně i vyřešit problémy s nedostatky dat, a zároveň lze této metody využít i když je dostatek dat. Přeskočení potřeby pomalé a náročné manuální anotace může být značné zjednodušení, v některých případech může být efektivnější investovat na začátku a vytvořit skript podobný mému, než manuálně anotovat velké datové sady.

Další výhoda je flexibilita a kontrola nad vytvořenými daty. Na rozdíl od reálných dat, která mohou být omezeny nebo zkresleny určitými proměnnými, jako jsou například špatné světelné podmínky, či nedostatečně vyškolený personál pořizující reálná data, tak syntetická data umožňují přesné řízení různých faktorů, jako je osvětlení, pozadí, pozice objektů apod. Tato kontrola vede k lepšímu modelování různých specifických scénářů a různorodých podmínek.

4.2 Jak lze na práci navázat a doporučení

V mé práci jsem neprozkoumal ani zdaleka všechny věci. Jako studentovi mi dělaly největší potíže právě čekání na vykreslení všech snímků a trénování. Aby jste věděly úspěšnost vašeho modelu, tak často potřebujete vymezit pár hodin na vykreslení všech snímků, a par dalších hodin na nějaké rychlé natrénování modelu. Proto pro efektivní tvorbu jsou potřebné plánovací schopnosti.

[1] Zajímavý článek o nedostupnosti reálných dat ve specializovaných odvětvích: linkedin.com/pulse/using-cgi-synthetic-imagescene-creation-train-computer-sheffer

Pro studenty nebo případně další, kteří by chtěli pracovat na podobné práci, doporučuji si dělat seznam věcí, které v nejbližších dnech potřebujete udělat, nezkoušejte jenom jednu věc a poté druhou, paralelizujte si všechno co jde. Pokud Vám na grafické kartě běží vykreslování nebo trénování, tak mezitím si organizujte vše potřebné. Připravte si grafy na vyhodnocení výsledků, opravte a automatizujte co vás naposledy zpozdilo. Program musí být stabilní, nesmí se jen tak stát, že uprostřed noci havaruje a několik hodin práce se ztratí. Tyhle věci mi dělaly v průběhu vývoje značné problémy. Ale i při dobrém postupu mě značně zdržovalo čekání na vytvoření datových sad či na trénování modelů.

Právě proto si myslím, že by bylo zajímavé prozkoumat jiné metody, které dokáží vykreslit snímky třeba i v reálném čase. Unreal Engine nebo i Unity jsou skvělé programy s výbornými komunitními obsahy a především fóry. V mé práci bylo obtížné nalézat řešení různých problémů, jelikož málo lidí využívá mojí kombinaci Octane, Blenderu a skriptování, každopádně u jiných programů tohle může být jinak. Dokonce by bylo možné i využít fyzické simulace a různé kombinace pro ještě více různorodé a robustní data.

4.3 Hledisko technického vývoje

V současném technickém vývoji je silný výpočetní výkon stále více a více dostupnější. Proto si myslím, že tenhle přístup může mít v budoucnosti dobrou šanci, že se stane více dostupnějším než sbírání reálných dat.

To neplatí nejen pro moji konkrétní metodu, ale pro všechny metody vytváření syntetických dat. S vývojem hardwaru se také vyvíjí software, s novými technologiemi, či starými technologiemi, které ale nebyly v minulosti dostupné kvůli výpočetní náročnosti. I dříve zmiňovaný Unreal Engine dokáže využít sofistikované technologie jako je sledování paprsků v reálném čase. Je tedy možné, že bude v blízké budoucnosti možno na vyžádání vytvořit syntetické datové sady s deseti tisíci snímky v rozmezí pár hodin – mnohem rychleji, než je možno nasbírat a anotovat reálné data, a s méně lidmi. Tohle by nebylo pouze efektivnější z časového hlediska, ale také i finančního.

4.4 Škálovatelnost

Mé řešení není dobře škálovatelné, jelikož se zaměřuje na velmi specifickou problematiku, každopádně programy jako Kubric jsou dostatečně robustní a škálovatelné. I dnes si lze pomocí ShapeNet a Kubric vytvořit syntetické data, bez potřeby modelování (s předpokladem, že daný objekt je dostupný na ShapeNet). Dostatečné kvalitní program by umožňoval interaktivní ovládání pro změnu podmínek – například kdyby jste si všimli špatného výkonu při určitých podmínkách, tak u dostatečně sofistikovaného programu by stačilo změnit pár parametrů, vytvořit dodatečné data, a tím zacelit nedostatky. Proto si myslím, že metody na vytváření syntetických dat se stanou čím dál více populární.

Kapitola 5

Implementace a výsledky

5.1 Generace syntetických dat pomocí Blender API skriptu

Každá datová sada má konfigurační JSON soubor. Zde se nachází hodnoty parametrů, obecné informace, ale také informace o již vytvořených snímcích. Tímto je bezproblémové, když se násilně vypíná skript. Po znovu-spuštění skript spolehlivě opět začne tam, kde skončil. Tento konfigurační soubor lze jednoduše vytvořit pomocí mého GUI programu, který jsem již ukazoval na obrázku 3.2.

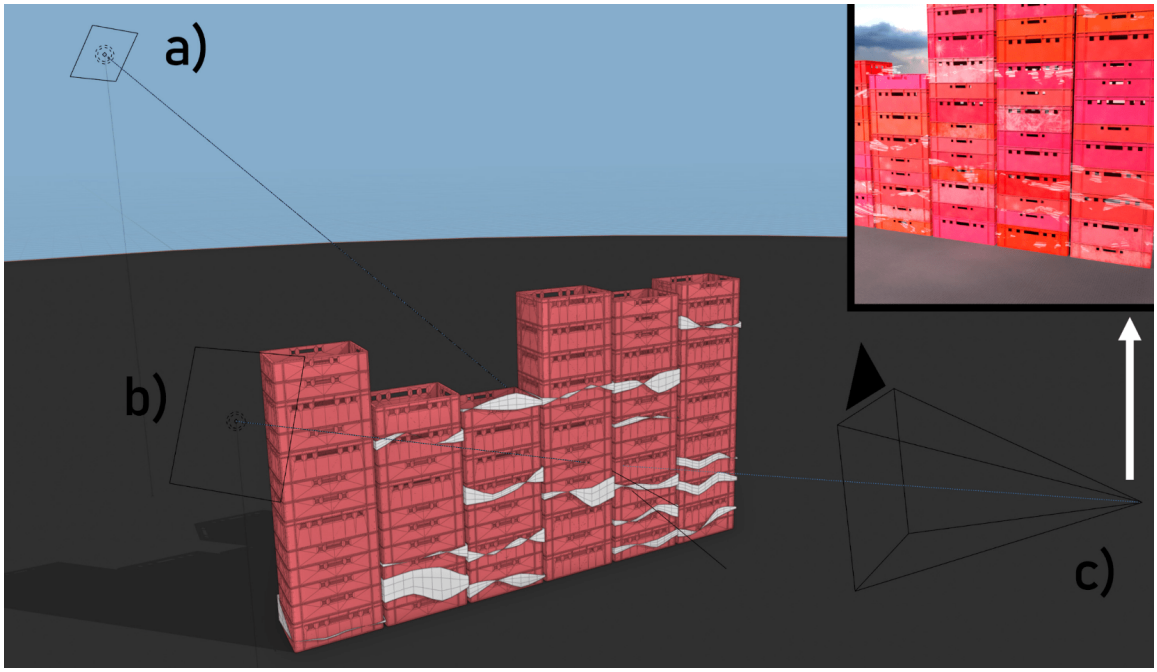
Anotace se ukládají po částech, dle výchozí hodnoty po každých 100 snímcích, a po dokončení všech snímků se spojí dohromady. Tímto se zefektivní práce, jelikož program nemusí pracovat s velkými proměnnými, ve kterých jsou všechny tato data uložena. Bez tohoto mechanismu by byly větší datové sady téměř nemožné.

Program podporuje přizpůsobení výsledných snímků pomocí 12 parametrů scény. Dále lze dodat vlastní textury a HDRI (High Dynamic Range Image) mapy do příslušných složek a program je začne automaticky využívat. Lze také změnit kolik přepravek se ve scéně vytvoří, každopádně tohle je poměrně archaické a využíval jsem tenhle mechanismus pouze na začátku vývoje, poté jsem používal pouze výchozí hodnoty.

Pro více pokročilé změny lze také změnit nastavení pro vykreslování, jako je například již zmíněný počet snímků, po kterém se rozdělí anotační soubory, interval restartu scény (další optimalizace), kvalita a formát obrázku, adaptivní vzorkování a jeho nastavení, a hlavně počet vzorků. Tato poslední možnost je důležitá, více vzorků znamená lepší kvalitu snímku, každopádně značně ovlivňuje dobu potřebnou na vykreslení. Výchozí hodnota je 200, která je hodně nízká, každopádně v mých testech nevytvářela značné artefakty, a když je vytvářela, tak pouze na podlaze při vysoké intenzitě světla, a to pod určitými úhly mezi kamerou a sluncem.

Skript sám ukončuje a spouští Octane Server, jelikož pravděpodobně úniky paměti způsobují, že server přestane odpovídat. Tohle jsem vyřešil pomocí watchdog procesu, který periodicky kontroluje, jestli skript vytvořil nové snímky. Pokud ne, tak restartuje Octane Server a znova zapne skript. Je ale samozřejmě předpoklad, že Octane server už byl zprovozněný, tedy uživatel se přihlásil (stačí si vytvořit účet zdarma, Octane pro Blender má velmi štědrou zkušební verzi).

Pro lepší čitelnost a souvislost jsem popsal jednotlivé parametry scény (intenzita osvětlení, výška kamery a podobně) v kapitole 5.2.2, která zároveň popisuje výsledky první fáze testů právě těchto parametrů.



Obrázek 5.1: Popis některých elementů ve výsledné scéně. Obrázek je jen ilustrační, jedná se o snímek z programu Blender s obarvenými objekty pro lepší srozumitelnost. Modrá je obloha (HDRI), šedá je podlaha, červená jsou přepravky a bílá jsou fólie. a) a b) jsou plošné světla, c) je kamera scény. Obrázek vpravo nahoře je výsledný snímek (z pohledu kamery scény). Všechny plošné světla a kamera sledují bod [0, 0, 0.75] ve středu snímku. Slunce je zhruba nad celou scénou a lze vidět stíny za přepravkami, které způsobuje jeho svit.

5.1.1 Popis scény

Jak vypadá scéna v editoru lze vidět na obrázku 5.1. Scéna není příliš komplexní, má přepravky, které se generují dle zvolených parametrů, a tvoří sloupce a řádky. Dále u každé přepravky je možnost, že se na ní vytvoří fólie. Materiál přepravky i materiál fólie využívá náhody pro vytváření rozmanitých kombinací, některé tyto elementy se dají ovlivnit pomocí změny parametrů scény. Dále se využívají HDRI[1] mapy na vytvoření denního osvětlení. Octane z této textury vytvoří oblohu, která neslouží pouze jako pozadí, ale také zde nalezne nejsilnější bod, a promění ho ve slunce a tedy i zdroj osvětlení. Sluneční svit poté směřuje z této oblasti oblohy. Následují dodatečné světla, dle parametrů a náhody se vytvoří 0 až 4 těchto světelných objektů na náhodných pozicích a s náhodnou intenzitou. Tyhle světla jsou čtverce o rozměru jednoho metru, které jsou natočené na střed scény, kde se také zhruba nachází střed všech přepravek. Dále se vytvoří kamera, která rovněž vždy sleduje střed scény. Více podrobných detailů o některých elementech, ukázek materiálů, a podobně se nachází v kapitole 5.2.2. Na referenčním stroji který reprezentuje poměrně zastaralý hardware (GTX 1060 6GB, i5-6600K, 16GB RAM) se v průměru jeden snímek vytvoří za 11 až 18 sekund dle nastavení, v případě vytvoření hodně fólií a silného světla se tohle zpomaluje. Z tohoto času se přibližně 40 % stráví vytvořením samotné scény, zbytek se stráví vykreslením snímku. Tohle platí pro výchozí nastavení a pro referenční stroj.

[1] HDRI mapy ve vysoké kvalitě, bezplatné, a často s CC0 licensí, jsou dostupné z: polyhaven.com/hdris/

5.2 Trénování modelů, popis parametrů scény a nalezení ideální kombinace

5.2.1 Volba hyperparametrů pro trénování

Před samotnými testy parametrů datové sady je zapotřebí zvolit správné hyperparametry na trénování. Zaručení konzistence mezi jednotlivými experimenty je kritické, například při malém počtu epoch a velké hodnoty rychlosti učení by mohl být značný rozdíl mezi modely, i přesto že byly trénovány na identické datové sadě. V tom případě by byly výsledky zavádějící a zcela nepoužitelné. Jelikož trénování není deterministické (například kvůli stochastickému gradientovému sestupu), tak nelze nikdy zaručit zcela konzistentní výsledky i při stejné datové sadě a stejných podmínkách.

Mým plánem bylo vytvořit experimenty, které prozkoumají kombinace hyperparametrů, a naleznou tu správnou kombinaci, která zajistí konzistentní výsledky, a také není výpočetně náročná. Tento druhý bod je důležitý, jelikož v první fázi testů parametrů plánuji vytvořit 3 instance pro všechny experimenty, dohromady budu celkem potřebovat natrénovat 231 modelů jen v první fázi. Proto je snaha najít tu správnou kombinaci, kdy modely začnou konvergovat co nejdříve. Kdyby jsem nalezl kombinaci, která by mi ušetřila pár desítek epoch na každý model, tak se celkový potřebný čas na natrénování všech 231 modelů značně sníží.

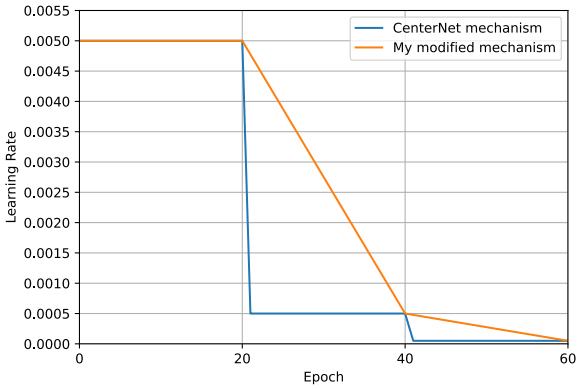
Pro nalezení té ideální kombinace hyperparametrů jsem vytvořil celkem 30 experimentů, které kombinovaly různé hodnoty rychlosti učení a snižování rychlosti učení. CenterNet pro útlum rychlosti učení používá skokový mechanismus, který zmenší současnou hodnotu rychlosti učení na faktor 10, když se dosáhne zadané epochy.

Výsledky nebyly moc uspokojivé. Problémem skokového snižování rychlosti učení je, že se úspěšnost modelu může zaseknout v lokálním minimu. V případech, kdy je vysoká hodnota rychlosti učení a sníží se na faktor deseti moc brzo, tak to může způsobit velmi odlišné výsledky od jednotlivých instancí trénování, i přes stejné hyperparametry a při stejné datové sadě, jelikož optimalizační proces nedokáže správně konvergovat nebo se může zaseknout v lokálním minimu. Tímto způsobem může skokové snižování rychlosti učení vést k nekonzistenci výsledků mezi jednotlivými tréninkovými běhy a může ovlivnit stabilitu a úspěšnost trénování. Je tedy podstatné zvolit vhodnou metodu snižování rychlosti učení, v mém případě byla nedostačující.

Proto jsem upravil výchozí mechanismus snižování rychlosti učení od CenterNet. Nový mechanismus snižuje hodnotu postupně, a obdobně jako původní mechanismus se hodnota mezi kroky sníží na faktor 10. Příklad, jak změna vypadá, lze vidět na obrázku 5.2 a v pseudokódu 10.

Jako další jsem upravil testovací skript. Předtím fungoval na primitivní bázi detekce, tedy testoval jsem, jestli se střed predikce nacházel v některé z referenční pravdě[2] (ground truth). V případě že ano, predikce se vyhodnotila jako pravdivá a referenční pravda se vymazala, v případě že ne, tak se predikce vyhodnotila jako nepravdivý pozitiv. Nijak jsem neošetřil případy, kdy se referenční pravdy protínají, takže pokud byla predikce přímo na průniku dvou nebo více referenčních pravd, tak bylo víceméně náhodné, která referenční pravda se správně predikovala, což mohlo způsobovat nepřesnosti. Proto jsem na další experimenty vytvořil sofistikovanější testovací skript, který funguje na bázi průniku přes sjednocení (Intersection over Union, často zkracováno na IoU), který je detailněji vysvětlen

[2]Referenční pravda je manuální anotace v testovací sadě. Při testování se zkouší, jestli predikce nesouhlasí s některou s referenčních pravd.



Obrázek 5.2: Ukázka průběhu starého a nového mechanismu snižování rychlosti učení. Příklad je při krokování v epochách 20 a 40.

Algorithm 1 Gradual Learning Rate Decay

```
steps ← selected_steps
steps.append(last_epoch)                          ▷ Add last epoch as final step
if epoch > min(steps) and epoch ∉ steps then
    start, end ← findClosest(steps, epoch)        ▷ Find closest steps to current epoch.
    value ← steps.index(start) + 1
    factor ← 10^(−value)
    start_lr ← initial_learning_rate × factor × 10
    end_lr ← initial_learning_rate × factor
    new_lr ← start_lr + (end_lr − start_lr) × ((epoch−start)/(end−start))   ▷ Linear interpolation
end if
```

v kapitole 3.2.1. Tenhle skript jsem používal až do konce práce, jen v případě potřeby jsem měnil testovací sadu.

Následně jsem udělal další testy. Použil jsem jednu stejnou datovou sadu s 200 syntetickými snímky. Z předešlých neúspěšných testů jsem alespoň našel vhodnou hodnotu pro rychlost učení, která měla dobrou vyváženost mezi úspěšností, ale nebyla tak moc velká, aby způsobovala zaseknutí na lokálním minimu. Každopádně předpoklad nového mechanismu snižování rychlosti učení je, že by se trénování již nemělo tolik zasekávat. Zároveň jsem nastavil počet epoch na vysokou hodnotu 200, aby jsem pro každý experiment našel v jaké epoše začne konvergovat, a tím tedy vyhodnotil, jak moc mohu zkrátit trénovací dobu bez ovlivnění úspěšnosti a konzistence mezi jednotlivými instancemi.

Tentokrát jsem použil stejnou hodnotu rychlosti učení a jenom jsem měnil kroky pro mechanismus snižování rychlosti učení. Předpoklad je, že čím později se mechanismus zapne, a čím mírnější je krokování, tím lepší výsledky, ale po delším čase. Cílem je tedy nalézt dostatečné konzistentní výsledky po co nejkratším čase. Vyzkoušel jsem 6 kombinací krokování, u každého jsem udělal 5 instancí pro prověření konzistence. Je důležité upozornit, že cílem není najít správné hyperparametry pro co největší úspěšnost, ale o co nejvíce konzistentní výsledky, jelikož pro nalezení těch ideální parametrů scény není podstatná absolutní úspěšnost, ale jen relativní úspěšnost mezi experimenty toho stejného parametru. Jelikož datové sady mají pouze 200 syntetických snímků a nepoužívá se augmentace, tak ani neočekávám vysokou úspěšnost.

Pro kontrolu konzistence je důležité vypočítat chybové odchylky. Budu používat metriku SEM[1] (Standard Error of Mean, česky standardní chyba průměru či standardní chyba střední hodnoty), kterou budu vypočítávat pro metriku F1-skóre. Potřebné vzorce pro výpočet SEM jsou:

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}} \tag{5.1}$$

$$\tag{5.2}$$

kde s představuje směrodatnou odchylku vzorku, x_i jsou jednotlivá pozorování ve vzorku, $\bar{x}$ je průměr vzorku a n je počet pozorování ve vzorku. Směrodatná odchylka je míra variability dat ve vzorku. Dále:

$$SEM = \frac{s}{\sqrt{N}} \tag{5.3}$$

$$\tag{5.4}$$

kde SEM představuje standardní chybu průměru, s je směrodatná odchylka vzorku a N je počet pozorování ve vzorku, v tomhle kontextu počet instancí, tedy 5.

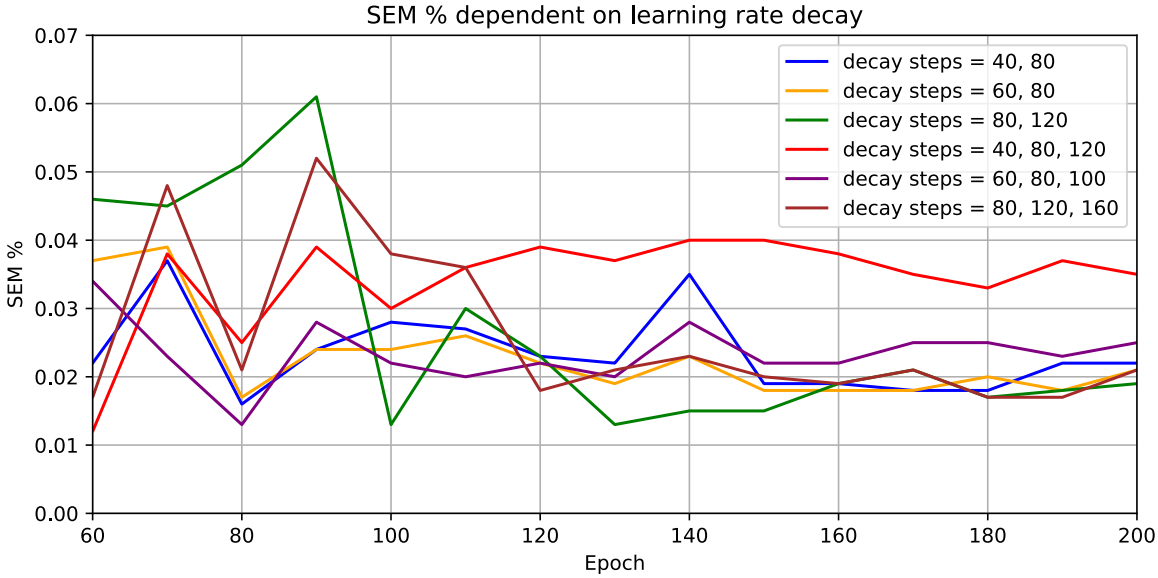
$$SEM_{\%} = \frac{SEM}{\bar{x}} \tag{5.5}$$

kde $SEM_{\%}$ představuje standardní chybu průměru vyjádřenou v relativních procentech, SEM je standardní chyba průměru, $\bar{x}$ je průměr vzorku.

Pomocí $SEM_{\%}$ lze porovnávat relativní procentuální chybu i mezi různými hodnotami vzorků. Například při hodnotách F1-skóre 0.1, 0.15 a 0.2 je procentuální chyba mnohem větší než u hodnot 0.5, 0.55 a 0.6. Převedením na relativní procenta lze porovnávat míru chyb u obou hodnot.

V grafu 5.3 jsou výsledky šesti experimentů s odlišným krokováním pro mechanismus snižování rychlosti učení. U každého experimentu jsem vytvořil pět instancí, a z nich jsem vypočítal jejich chybovou odchylku. Je zapotřebí najít co nejmenší a zároveň co nejvíce konzistentní hodnoty. Zde jsou slovní popisy těchto výsledků:

- 40, 80: Poměrně dobré výsledky, i přesto, že se rychlost učení aktivovala poměrně brzo. Také byl výrazný skok v epoše 140.
- 60, 80: Po 90 epoše se značně stabilizoval a poté se držel u nejnižších hodnot. Hodnota se poměrně dobře stabilizuje už v epoše 90.
- 80, 120: Nejhorší výsledky před epochou 100. Značné skoky v epochách 90, 100 a 110.
- 40, 80, 120: Když se zahrnou všechny epochy tak je překvapivě stabilní, každopádně téměř vždy nad hodnotou 3 %. Pro dobré výsledky by bylo zapotřebí trénovat na alespoň 120 epochách.
- 60, 80, 100: Podobný jako experiment s 60, 80, ale o zhruba 0.5 % vyšší hodnoty. Hodnota se stabilizovala po hodnotě 100.



Obrázek 5.3: Experimenty s odlišným krokování pro snižování rychlosti učení. Hodnotí se hodnota SEM %, neboli chybová odchylka v procentech. Nižší a stabilnější hodnoty jsou lepší. V grafu nejsou zobrazeny epochy 1 až 59. Je vyobrazena pouze každá desátá epocha.

- 80, 120, 160: V epoše 90 značný skok, ale po epoše 120 se značně stabilizuje. Po epoše 120 se stabilizuje.

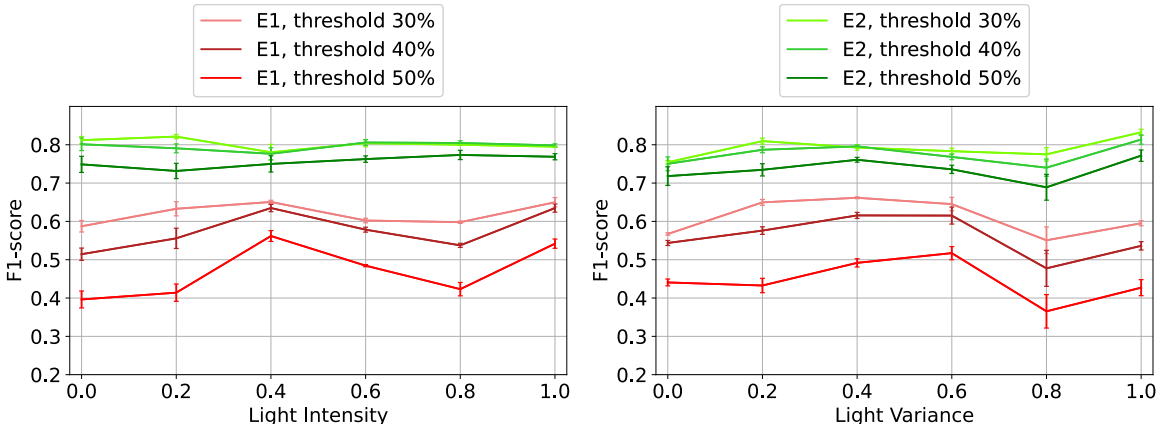
Až na pár výjimek, většina experimentů měla dobré výsledky, takže jsem se rozhodoval hlavně dle toho, kde se dosáhne dobré hodnoty v co nejdřívější epoše. Nakonec jsem zvolil hodnotu krokování 60, 80, jelikož měla poměrně dobré výsledky nejen v pozdních epochách. Zvolil jsem tedy tohle krokování, s tím, že budu trénovat až do epochy 100.

5.2.2 Fáze 1: Obecné prozkoumání hodnot parametrů scény a jejich podrobný popis

V této kapitole rozebírám jak fungují parametry scény a výsledky experimentů, které je měnily. Pro lepší čitelnost jsem v pseudokódech vynechal obsah jednoduchých funkcí, konkrétně `randSign()`, která náhodně vrátí hodnotu -1 nebo 1, `randInt(x, y)`, která vygeneruje celé číslo v daném rozsahu včetně, a jako poslední `randFloat(x, y)`, která generuje reálné číslo v daném rozsahu včetně. Hodnota parametrů je již inicializována na začátku kódu. Tyto poznámky platí pro pseudokódy 2, 3 a 4. Pro vyhodnocování jsem použil metriku F1-skóre. Bral jsem v potaz hlavně její průměrnou hodnotu napříč všemi prahy a oběma přepravkami, v případě, že hodnoty byly blízko sebe, tak jsem se řídil jednotlivými hodnotami. Tyto data jsou k dispozici jako Excel tabulky v přiložených souborech. F1-skóre a úspěšnost je v kontextu této kapitoly to stejné.

Parametry světla

V obrázku 5.4 lze vidět úspěšnost parametrů intenzity a variace světla. Parametr intenzity světla ovlivňuje počet vytvořených světel a zároveň výslednou intenzitu světla. Variance světla, podobně jako jeho intenzita, ovlivňuje počet i intenzitu vytvořených světel. Variace dále ovlivňuje pozici světla, tedy čím je větší variance, tím je větší šance že se vytvoří



Obrázek 5.4: Grafy pro přepravky E1 a E2 s metrikou F1-skóre pro parametry intenzity a variace světla. Legendy patří k oběma grafům.

buďto dále než obvykle, nebo blíže než obvykle, zatímco při malé variaci se vytvoří ve více konzervativní pozici. Variace dále ovlivňuje výslednou intenzitu světla. Detailnější popis, jak tyhle dva parametry ovlivňují scénu, je v pseudokódu 2. Všimněte si, že orientace světel je nastavena na sledování bodu ve středu scény, to je proto, jelikož Octane využívá reálný přístup k světlu. Každý světelný objekt má nějakou plochu, ze které vyzařují paprsky, v případě těchto světel se jedná o čtverce o rozměru 1 metru, které vyzařují světlo jedním směrem, tedy ke středu scény. Jedná se o více sofistikovanější přístup než často využívaný nekonečně malý bod, který vyzařuje světlo všude kolem sebe, jelikož tohle není v reálném světě možné.

Algorithm 2 Create Area Lights

procedure SETUPAREALIGHTS
  $variance \leftarrow 1 + \text{RANDSIGN} \times \frac{variance}{4}$ ▷ Adjust light variance
  $max\_light\_count \leftarrow \lfloor 4 \times \text{intensity} \times \text{variance} \rfloor$ ▷ Max number of lights
  $N \leftarrow randInt(0, max\_light\_count)$ ▷ Pick number of lights to create
  repeat N **times**
    CREATEAREALIGHT
  end
end procedure
procedure CREATEAREALIGHT
  $intensity \leftarrow 0.5 + \text{intensity}$ ▷ Adjust intensity
  $variance \leftarrow 1 + \text{RANDSIGN} \times \frac{variance}{2}$ ▷ Adjust variance
  $x \leftarrow \text{variance} \times \text{RANDFLOAT}(-10, 10)$
  $y \leftarrow \text{variance} \times \text{RANDFLOAT}(-10, 10)$
  $z \leftarrow \text{variance} \times \text{RANDFLOAT}(5, 15)$ ▷ Z is up/down axis
  $light \leftarrow \text{createLight}(location = [x, y, z])$ ▷ Create light object at position
  $light.\text{TRACKTARGET}([0, 0, 0])$ ▷ Point light at the origin
  $variance \leftarrow \text{RANDFLOAT}(0.5, 2) \times \text{variance}$ ▷ Adjust variance
  $light.value \leftarrow \text{variance} \times \text{intensity}$ ▷ Set light intensity
end procedure

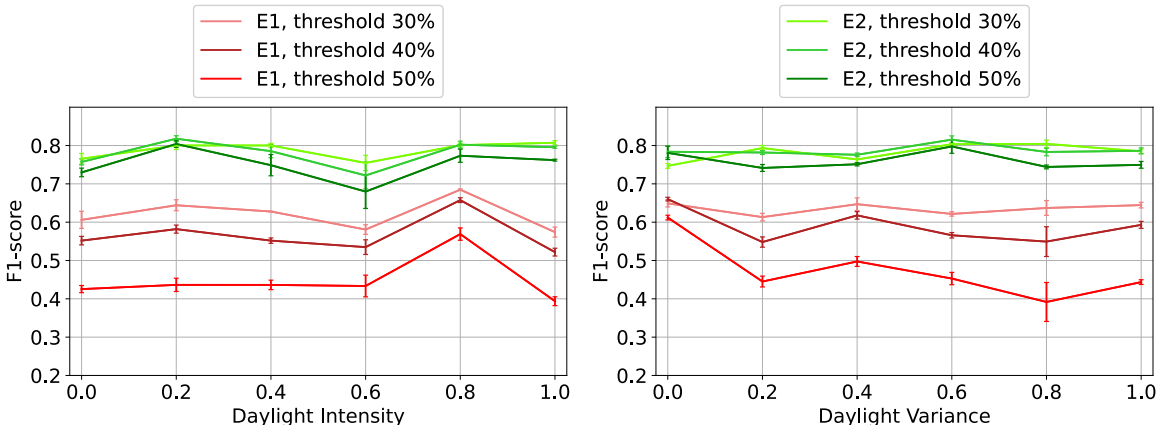
Úspěšnost při změně intenzity světla se u přepravky E2 nijak značně nemění, každopádně pro přepravku E1 je v hodnotě 0.4 lokální maximum, zatímco v hodnotě 0.8 je propad, ale poté je v hodnotě 1.0 globální maximum. Vybral jsem tedy hodnotu 1.0 která měla

průměrné F1-skóre 0.6981 jako novou výchozí hodnotu, a hodnotu 0.4 s výsledkem 0.6924 dále otestuji.

Ve variaci světla je propad v hodnotě 0.8. Tato hodnota měla zároveň velkou hodnotu chybové odchylky, dokonce největší ze všech provedených experimentů, tedy 6.4 %. Každopádně průměrná hodnota napříč oběma přepravkami by se i při menší chybové odchylce ani z daleka nerovnala nejúspěšnějším hodnotám, proto jsem tuhle hodnotu ignoroval.

Hodnoty 0.2, 0.4 a 0.6 měli největší úspěšnost a byly srovnatelné, konkrétně s úspěšnostmi 0.6649, 0.6864 a 0.6775 a chybová odchylka byla u všech pod 2.2 %. Hodnotu 0.4 jsem vybral jako novou výchozí hodnotu pro variaci světla, a hodnoty 0.2 a 0.6 dále otestuji v druhé fázi.

Parametry denního osvětlení



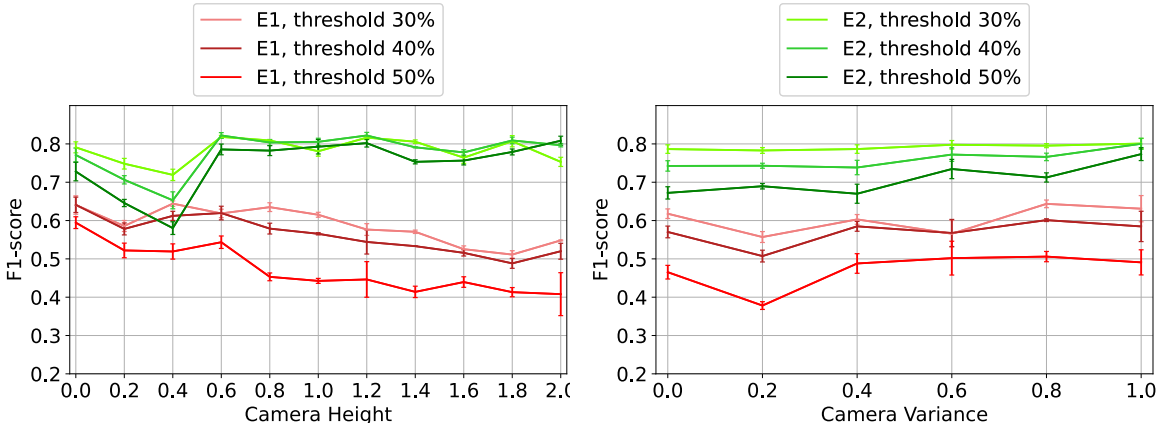
Obrázek 5.5: Grafy pro přepravky E1 a E2 s metrikou F1-skóre pro parametry intenzity a variace denního osvětlení. Legendy patří k oběma grafům.

Výsledky pro experimenty s parametry scény pro denní světlo lze vidět v obrázku 5.5. Denní světlo se vytvoří z HDRI textury. Nalezne se nejsilnější bod a z něj se vytvoří slunce. V programu je k dispozici celkem 10 odlišných HDRI, pro každou scénu se náhodně vybere jedna. Tyto snímky kromě vytvoření slunečního svitu také vytvoří pozadí, tedy oblohu. Parametry ovlivňují sílu tohoto osvětlení, tak jak by jste očekávaly, vyšší variace vytvoří více extrémnější podmínky více často, zatímco intenzita tenhle efekt zesílí. Pokud budou obě hodnoty vysoké, tak to způsobí, že se často vytvoří snímky s velice silným, nebo naopak velice slabým denním osvětlením. V opravdu extrémních případech je denní světlo tak slabé, že scénu a přepravky lze sotva rozeznat. Je předpoklad že tohle sníží úspěšnost, a proto tyhle extrémní hodnoty parametrů by se neměly dostat do další fáze. Konkrétní vzorce, jak se denní světlo vypočítá, lze vidět v pseudokódu 3.

Oba parametry měli individuálně jen jednu jasnou nejlepší hodnotu, konkrétně 0.8 pro intenzitu denního světla a 0 pro variaci denního světla. Obě tyhle hodnoty byly o více než 0.025 lepší než druhé nejlepší hodnoty. Zvolil jsem je tedy jako nové výchozí hodnoty do druhé fáze.

Algorithm 3 Create Daylight

procedure SETUPDAYLIGHT
  $intensity \leftarrow \text{intensity} + 0.5$ ▷ Adjust daylight intensity
  $variance \leftarrow 1 + \text{RANDSIGN} \times \frac{\text{variance}}{2}$ ▷ Adjust daylight variance
  $variability \leftarrow \text{RANDFLOAT}(0.75, 2) \times \text{variance}$
  $daylight.power \leftarrow \text{variability} \times \text{intensity}$ ▷ Set daylight intensity
  $daylight.sun\_size \leftarrow \text{RANDFLOAT}(0.75, 1.25) \times \text{variance}$ ▷ Set size of the sun
end procedure



Obrázek 5.6: Grafy pro přepravky E1 a E2 s metrikou F1-skóre pro parametry výšky a variace kamery. Legendy patří k oběma grafům.

Parametry kamery

Úspěšnost parametrů kamery lze vidět na obrázku 5.6. Parametr výšky kamery ovlivňuje pouze samotnou výšku, dále variace kamery ovlivňuje polohu kamery na všech osách, podrobné vzorce lze vidět v pseudokódu 4.

Algorithm 4 Create Camera

procedure SETUPCAMERA
  $height \leftarrow \text{height} + 1$ ▷ Adjust height
  $variance \leftarrow 1 + \text{RANDSIGN} \times \frac{\text{variance}}{2}$ ▷ Adjust variance
  $x \leftarrow \text{RANDFLOAT}(-3, 3) \times \text{variance}$
  $y \leftarrow \text{RANDFLOAT}(-4, 1) \times \text{variance}$
  $z \leftarrow \text{RANDFLOAT}(0, 4) \times \text{variance} \times \text{height}$ ▷ Z is up/down axis
  $camera.location \leftarrow (x, y, z)$ ▷ Set camera location
end procedure

Parametr výšky kamery byl problematický už ze začátku. Je to jediný parametr, u kterého jsem zpětně zvýšil meze na $\langle 0.0, 2.0 \rangle$, jelikož při počátečních pokusech nanečisto úspěšnost značně rostla u vyšších hodnot, takže jsem je chtěl prozkoumat. Každopádně skript na vytváření syntetických dat ještě nebyl zcela kompletní, a jak lze vidět, tak v současné době mají menší hodnoty větší úspěšnost. Zároveň jsem při testech v rané fázi používal pochybnou testovací metodu. Nakonec jsem tyhle věci opravil a tyhle problémy se potlačily, ale i přesto je tenhle parametr nejvíce sporadický, zejména u přepravky E2 lze vidět různé skoky v úspěšnosti. V grafech lze vidět, že výsledky u jednotlivých přepravek jsou do jisté míry svým opakem. Když úspěšnost E1 klesá, tak E2 roste, to může být způsobeno jiným tvarem

Obrázek 5.7: Příklad možné chyby při vysoké hodnotě variace kamery. Snímek nevytvořil žádnou anotaci, jelikož se kamera vytvořila moc blízko a nezabírala celou přepravku.

vrchní části přepravky. Přepravka E2 je větší a má více vyztuženou horní část, která může jinak vrhat stíny nebo se obecně jinak chovat než přepravka E1. Nakonec jsem se rozhodl použít hodnotu 0.0 jako výchozí a dále otestovat hodnotu 0.6, jelikož E1 je obtížnější na natrénování, a tímhle se více přiblíží úspěšnost jednotlivých druhů přepravek.

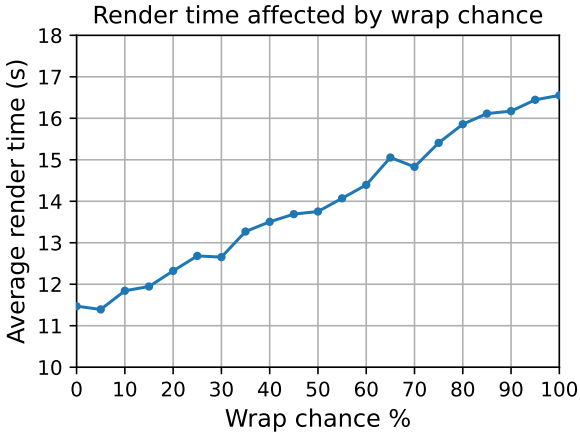
Variace kamery byla oproti výšce kamery přímočará, nejlepší hodnota byla 1.0, která postoupila do další fáze.

V obrázku 5.7 lze vidět možnou chybu při vysoké hodnotě variace kamery. Pokud je tato hodnota moc velká, tak ovlivní jak blízko (nebo daleko) se může kamera vytvořit, rovněž tato hodnota zvyšuje šanci, kdy se kamera vytváří v takovýchto extrémnějších hodnotách. Problém generování kamery příliš blízko přepravkám je ten, že skript nevytvoří anotaci pro přepravku, jelikož jsou body její segmentační masky mimo snímek. V tomhle konkrétním snímku se nevytvořila ani jedna anotace. Tohle nemá vliv na úspěšnost, každopádně stále se jedná o zcela zbytečný snímek, který k ničemu neslouží. Jedná se spíše o extrémní situaci, i při vysoké hodnotě variace se tohle stane zhruba v méně než 1 % snímků.

Parametry fólie

Parametr šance na vytvoření fólie reprezentuje šanci, že se spolu s jednotlivou přepravkou vygeneruje i fólie. Tato fólie je jednoduchý, předem vygenerovaný 3D objekt. Každopádně u tohoto parametru je problém s optimalizací, jelikož při vyšších hodnotách se zpomaluje vykreslování snímků, jak lze vidět na grafu 5.8. Rozdíl je značný, a proto jsem uznal za vhodné zde použít menší hodnoty, pokud přitom značně neutrpí úspěšnost.

Na obrázku 5.9 lze vidět materiál fólie. Je snaha vytvořit iluzi odlesku, jelikož skutečný odlesk by bylo obtížně vytvořit v náhodně vygenerované scéně, a také by to bylo výpočetně náročnější. Materiál má bílou barvu jako základ, zbytek materiálu vytváří alfa masku pomocí různých výpočtů a kombinací textur. Horní textura v obrázku slouží k vytvoření velkých oblastí fólie, které budou viditelné, je snaha napodobit situaci, kdy jen část fólie odráží světlo ze zdroje přímo to kamery. Druhá textura slouží k vytvoření jemnějších detailů,



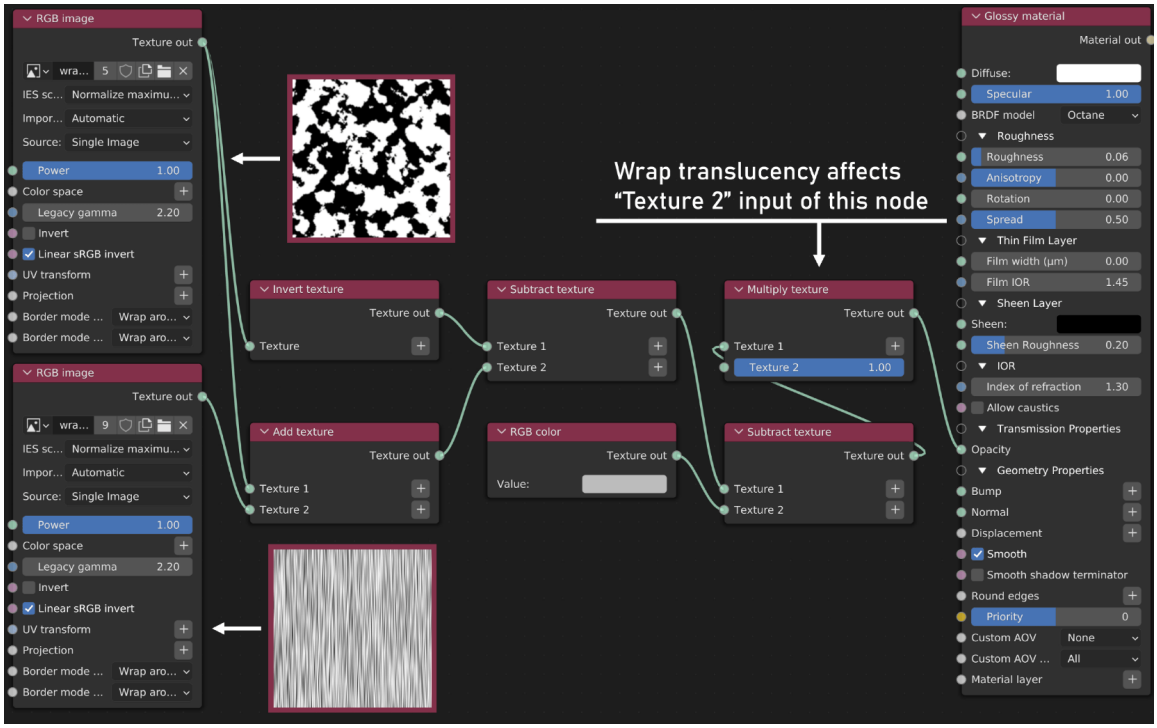
Obrázek 5.8: Graf závislosti parametru šance pro vytvoření fólie na průměrný čas potřebný k vykreslení snímku. Rychlost při hodnotě 100 % je o 44 % pomalejší než při hodnotě 0 %. Tato závislost je lineární. Kvůli tomu jsem preferoval menší hodnoty parametru i v případech, kdy měly mírně vyšší úspěšnost.

konkrétně jsou v této textuře pruhy, které se snaží napodobit záhyby natažené fólie. Oba tyhle typy mají více podobných textur, které se náhodně vybírají pro vetší rozmanitost.

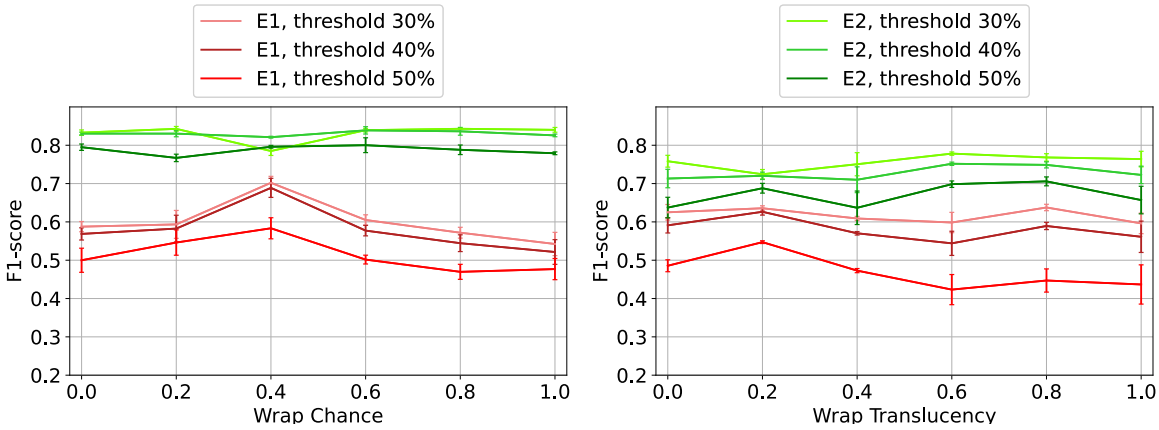
Samotný objekt fólie je jednoduchý 3D model, na který je aplikován tento materiál. Kvůli optimalizaci jsem vytvořil skript, který je předem vytvoří, a ve výsledném program jsem použil 1000 možných, před vytvořených modelů fólie pro přepravku E1 a dalších 1000 pro E2. Když se má vytvořit fólie, tak se vždy vybere jeden model náhodně. Jedná se se jednoduchý proces, kdy jsem připravil body na 3D souřadnicích, které jsou z venku přepravky, následně se náhodně mění souřadnice na bodech v ose Z (Blender využívá pravotočivou kartézskou soustavu, tedy osa Z je nahoru/dolů). Následně se vytvoří UV mapa a uloží se model. Tímto se vytvoří různé patvary, a pomocí materiálu, který je z velké částí průhledný, se vytvoří poměrně věrohodná napodobenina fólie s iluzí odlesku. Jak to vypadá v praxi můžete vidět na obrázku 2.4 nebo 2.5.

V grafech 5.10 lze vidět úspěšnosti obou parametrů fólie. Parametr šance na vytvoření fólie měl v hodnotě 0.4 nejlepší F1-skóre s 0.7294, hodnoty 0.2 a 0.6 byly o zhruba 4 % horší. Každopádně rozdíl průměrného času renderování mezi hodnotou 0.2 a 0.4 se mi nezdál jako dobrý kompromis, každopádně jsem nakonec zvolil méně úspěšnou hodnotu 0.2, i když v retrospektivě by jsem se možná rozhodl jinak,.

Parametr průsvitnosti fólie ovlivňuje poslední uzel v jejím materiálu, tento uzel násobí texturu. Před tím se ještě přičte k parametru hodnota 0.5. Tedy menší hodnota parametru značí větší průsvitnost, a větší znamená více matnou texturu. Jelikož se k hodnotě přičítá 0.5, tak výsledná hodnota může být více než 1. Jelikož materiál pracuje s alfa texturou pro vytvoření iluze odrazu světla od fólie, tak hodnota nad 1 může způsobit větší skok mezi částí která je zcela průhledná a částí která jen lehce průhledná, jelikož při násobení zcela průhledné částí se nic nezmění (násobení nulou dá vždy nulu), zatímco při násobení lehce průhledné části se část stane více matná.



Obrázek 5.9: Materiál fólie. Využívá dvě textury, které mění alfa masku, jedna textura je na vytvoření hlavních viditelných oblastí, a druhá na vytvoření detailů, tedy vytvoření iluze záhybů fólie.

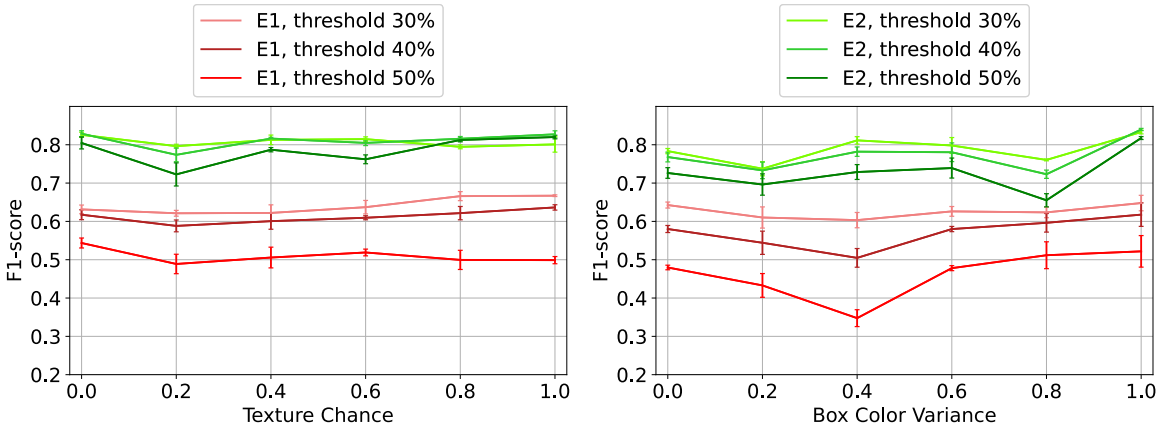


Obrázek 5.10: Grafy pro přepravky E1 a E2 s metrikou F1-skóre pro parametry šance na vytvoření fólie a průhlednosti fólie. Legendy patří k oběma grafům.

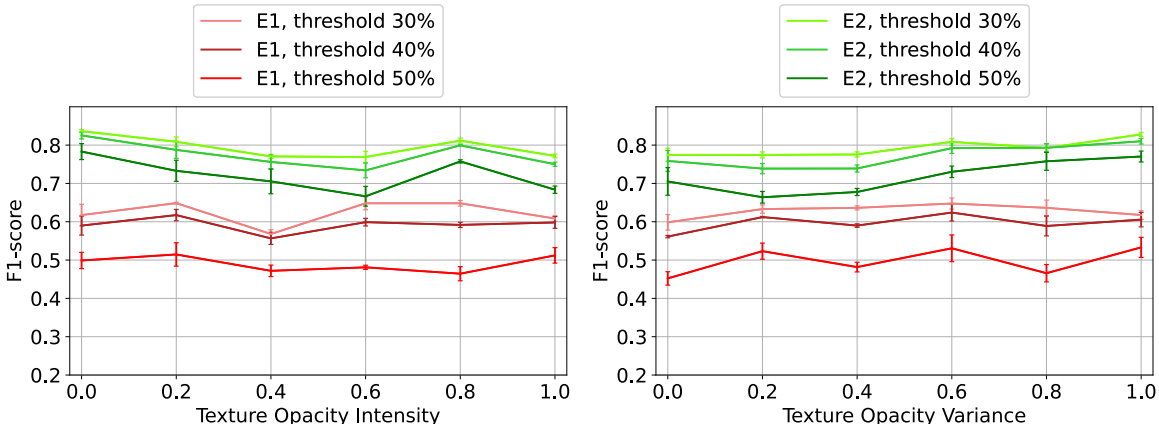
Parametry materiálů přepravek

V obrázcích 5.11 a 5.12 lze vidět výsledky parametrů variace průhlednosti textury, intenzity průhlednosti textury, šance na využití této textury, a nakonec variaci základní barvy přepravky. Všechny tyto parametry ovlivňují materiál přepravky.

Variace barvy přepravky se využije vždy, každopádně se jedná o poměrně slabý faktor. Maximální a minimální hodnoty jsou hodně mírné, nelze třeba vytvořit zelenou či modrou barvu, je pouze daná téměř čistě červená barva, a jednotlivé hodnoty kanálu RGB se jen



Obrázek 5.11: Grafy pro přepravky E1 a E2 s metrikou F1-skóre pro parametry šance na použití textury přepravky a variace základní barvy přepravky. Legendy patří k oběma grafům.



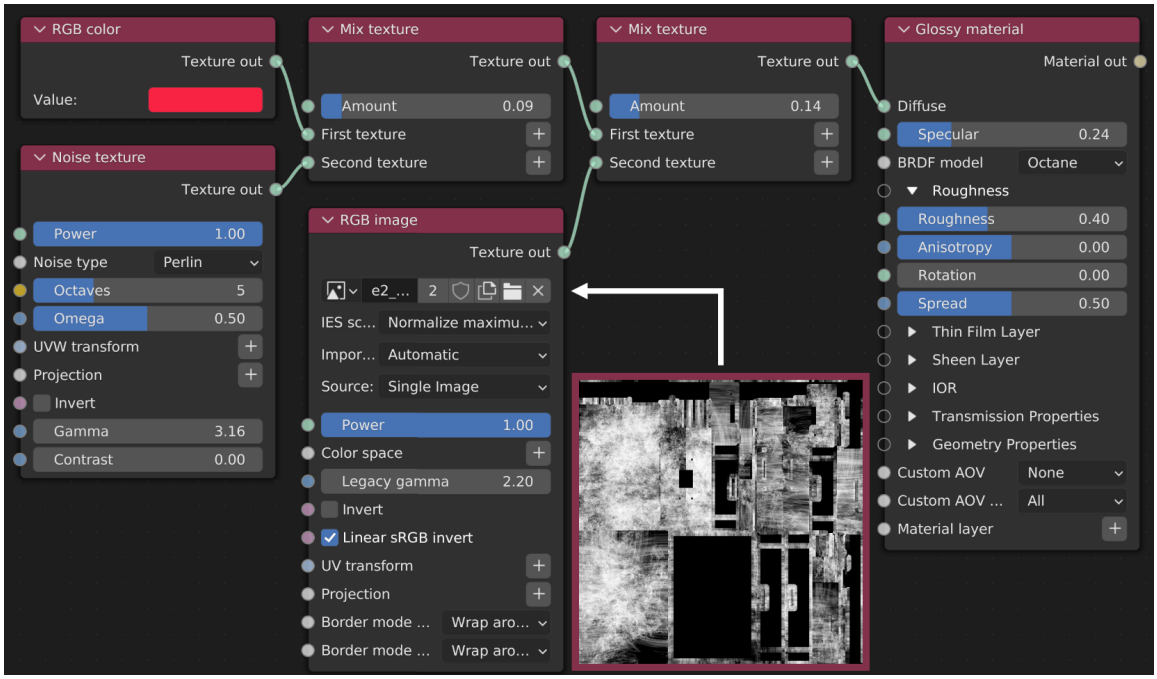
Obrázek 5.12: Grafy pro přepravky E1 a E2 s metrikou F1-skóre pro parametry intenzity a variace průhlednosti textury přepravek. Legendy patří k oběma grafům.

mírně a opatrně mění. Občas se také použije výrazně odlišná barva, která se snaží napodobit přepravky se silně degradovanou barvou jak v reálných snímcích, každopádně tohle není ovlivněno žádným parametrem, jedná se o odlišný proces. Nejlepší hodnota tohoto parametru byla hodnota 1.

Dalším parametrem je šance na vytvoření textury. V tomhle kontextu se jako texturou myslí obrázek, který se nanese na vrch materiálu, jedná se spíše o masku, která vytváří různé efekty opotřebení přepravky. Zajímavě, hodnoty 0 a 1 vyšly s téměř totožnou úspěšností. Jako výchozí hodnotu jsem zvolil 1 a hodnotu 0 dále otestuji.

Následně jsou parametry variace a intenzity průhlednosti právě této textury. Jak jejich názvy napovídají, tak mění, jak moc je tato textura průhledná. Pro variaci jsem zvolil hodnotu 1 jako výchozí a 0.6 na další testování, a pro intenzitu jsem jako výchozí zvolil hodnotu 0.2 a dále jsem dal hodnotu 0 na otestování.

Na obrázku 5.13 je vidět materiál přepravky. Začíná vytvořením základní barvy, tohle je ovlivněno parametrem scény variace barvy přepravky. Tahle barva se smíchá s náhodným šumem v mixovacím uzle. Tenhle šum není značně viditelný, jelikož úroveň mixování těchto dvou uzlů je náhodná hodnota mezi 0.05 až 0.1, takže v každém případě bude zá-



Obrázek 5.13: Materiál přepravky s použitou texturou. Některé hodnoty v uzlech jsou náhodně změněny pomocí náhody, v některých případech jsou ovlivněny i pomocí parametrů scény. Do uzlu *RGB image* je zapojena textura (je k dispozici více textur, vybírají se opět náhodně), která vytváří efekt opotřebení přepravky. Šance na použití tohoto uzlu je podmíněna stejnojmenným parametrem scény.

kladní barva mnohem více dominantní než šum. Dále se tohle mixuje se samotnou texturou, každopádně tahle textura se nevyužije vždy, tohle je ovlivněno parametrem scény *texture chance*. V případě, že se využije, tak se smíchá s dosavadním výsledkem. Poměr na smíchání je ovlivněn parametry scény *texture opacity variance* a *texture opacity intensity*. Následně se výsledek přidá do difuzní mapy materiálu. Dále se ještě nastaví úroveň hladkosti (jak moc materiál rozptýlí odražené světlo) v rozmezí 0.2 až 0.8, a úroveň odlesku v rozmezí 0 až 0.5. Podobně jako u textur fólie, je i zde více dostupným textur, na obrázku je jen jeden příklad. Tyto textury jsem vytvořil pomocí Adobe Substance Painter 3D[3]. Jsou zvlášť textury pro přepravku E1 a E2, snažil jsem se vytvořil textury připomínající různé opotřebení, zašpinění a případně i praskliny.

V tabulce 5.1 je kompilace těch nejúspěšnějších hodnot pro každý parametr. Tyto hodnoty budou použity jako výchozí hodnoty pro vytváření nových datových sad v fázi 2, v které se otestují další hodnoty, které byly blízko těm nejúspěšnějším.

5.2.3 Fáze 2: Prozkoumání nejslibnějších hodnot

Cílem druhé fáze bylo prozkoumat hodnoty, které sice nebyly nejvyšší pro daný parametr, každopádně byly dostatečně blízkou nejvyšší hodnotě. Jelikož výsledky mohou mít odchylky, tak je šance, že i tyto datové sadu mohou mít větší úspěšnost. Zároveň se mohou chovat jinak při nových výchozích parametrech, například jelikož parametry intenzita a variace osvětlení ovlivňují stejné elementy v scéně, tak je předpoklad, že jeden parametr se bude

[3]Adobe Substance Painter 3D: www.adobe.com/products/substance3d

Phase 2 default values

Light Intensity	1.0	Box Color Variance	1.0	Wrap Translucency	0.0
Light Variance	0.4	Camera Height	0.0	Texture Opacity Intensity	0.2
Daylight Intensity	0.8	Camera Variance	1.0	Texture Opacity Variance	1.0
Daylight Variance	0.0	Wrap Chance	0.2	Texture Chance	1.0

Tabulka 5.1: Seznam výchozích hodnot parametrů, které budou použity v fázi 2. Tyto hodnoty byly vybrány jako nejslibnější hodnoty parametrů v 1. fázi.

Dataset	Parameter	Value	F1-score						Average
			E1 30%	E1 40%	E1 50%	E2 30%	E2 40%	E2 50%	
EX800	Reference (default values)		0.7151	0.6840	0.6152	0.8025	0.7993	0.7607	0.7295
EX801	Light Intensity	0.4	0.6871	0.6821	0.6455	0.8360	0.8356	0.8090	0.7492
EX802	Light Variance	0.2	0.6203	0.6142	0.5781	0.8000	0.8161	0.8008	0.7049
EX803	Light Variance	0.6	0.6232	0.6030	0.5530	0.8183	0.8208	0.7971	0.7026
EX804	Camera Height	0.6	0.5859	0.5753	0.5256	0.8102	0.7881	0.7294	0.6691
EX805	Texture Opacity Intensity	0	0.6845	0.6648	0.6190	0.8166	0.8233	0.7945	0.7338
EX806	Texture Opacity Variance	0.6	0.6247	0.6068	0.5611	0.8227	0.8158	0.7847	0.7026
EX807	Texture Chance	0	0.5998	0.5942	0.5610	0.7785	0.7726	0.7396	0.6743

Tabulka 5.2: Tabulka výsledků druhé fáze. Testy probíhaly na hranicích 30%, 40% a 50%, v tabulce jsou pro lepší čitelnost sloučeny. Modrý řádek jsou výsledky modelu s výchozími hodnotami, zelená jsou modely které měly větší úspěšnost než referenční výchozí modely, zbylé experimenty měly nižší úspěšnost a byly tedy vyřazeny. Pro rozhodování jsem se řídil hlavně metrikou accuracy u obou typů přepravek. Hodnoty parametrů u více úspěšných datových sad se staly novými výchozími hodnotami, zbylé hodnoty zůstaly stejné.

chovat jinak když se změní druhý parametr, jelikož parametry na sebe navazují. Extrémní příkladem je třeba parametr na použití textury přepravky – pokud je nulový, tak parametry intenzity a variace průhlednosti této textury nic neovlivňují.

V tabulce 5.2 lze vidět výsledky experimentů druhé fáze. Ve dvou ze sedmi případů se nalezla lepší hodnota. Referenční datová sada experimentu EX800, tedy zvolená kombinace těch nejlepších výsledků z první fáze, měla F1-skóre téměř 0.73. Pro porovnání, průměrné F1-skóre pro experimenty v první fázi bylo 0.6638, tedy povedlo se mi zvýšit tuhle úspěšnost o téměř 10 %, čímž jsem si prověřil, že hledání těch správných parametrů opravdu mělo smysl. Kombinaci parametrů jsem aktualizoval na nové nejlepší hodnoty, které jsou v tabulce 5.3. Každopádně zde lze vidět jeden nepříjemný fakt – parametr scény *texture opacity intensity*, neboli jak moc matná bude výsledná textura, skončil s hodnotou 0. Jelikož se výsledná matnost textury násobí tímto číslem, tak to efektivně znamená, že tato textura se vůbec nebude využívat (technicky bude, ale bude zcela průsvitná). Je to škoda, jelikož vytvoření tohoto materiálu a všech textur nebyla jednoduché ani rychlé, každopádně výsledky hovoří jasně, při použití textury je menší úspěšnost. Tohoto si šlo všimnout i v první fázi, když hodnoty 1 i 0 měly porovnatelné výsledky.

5.2.4 Fáze 3 a 4: Zmenšování krokování a nalezení konečných hodnot parametrů scény

Fáze 3 a 4 mají za úkol prozkoumat okolní hodnoty dosud nejúspěšnějších hodnot parametrů scény s menším krokováním. Například pokud byla nejúspěšnější hodnota 0.4, tak třetí fáze prozkoumá hodnoty 0.3 a 0.5. V případě, že se 0.3 se prokáže jako nejlepší, tak ve čtvrté

Phase 3 default values					
Light Intensity	0.4	Box Color Variance	1.0	Wrap Translucency	0.0
Light Variance	0.4	Camera Height	0.0	Texture Opacity Intensity	0.0
Daylight Intensity	0.8	Camera Variance	1.0	Texture Opacity Variance	1.0
Daylight Variance	0.0	Wrap Chance	0.2	Texture Chance	1.0

Tabulka 5.3: Seznam výchozích hodnot parametrů, které budou použity v fázi 3. Tyto hodnoty byly vybrány jako nejslibnější hodnoty parametrů v 2. fázi.

Dataset	Parameter	Value	F1-score						Average
			E1 30%	E1 40%	E1 50%	E2 30%	E2 40%	E2 50%	
EX808	Reference (default values)		0.7112	0.7151	0.6682	0.8202	0.8156	0.7899	0.7534
EX809	Light Intensity	0.3	0.7371	0.7245	0.6788	0.8303	0.8279	0.7916	0.7651
EX810	Light Intensity	0.5	0.6233	0.5983	0.5228	0.7907	0.7728	0.7257	0.6723
EX811	Light Variance	0.3	0.6809	0.6755	0.6133	0.8382	0.8238	0.7883	0.7367
EX812	Light Variance	0.5	0.7519	0.7440	0.6962	0.8166	0.8258	0.8061	0.7734
EX813	Daylight Intensity	0.7	0.6276	0.6046	0.5556	0.8412	0.8354	0.7907	0.7092
EX814	Daylight Intensity	0.9	0.7270	0.7199	0.6590	0.8244	0.8309	0.8115	0.7621
EX815	Daylight Variance	0.1	0.6417	0.6233	0.5660	0.8441	0.8496	0.8228	0.7246
EX816	Box Color Variance	0.9	0.7045	0.7081	0.6715	0.8483	0.8536	0.8362	0.7704
EX817	Camera Height	0.1	0.6620	0.6514	0.5924	0.8041	0.8132	0.7749	0.7163
EX818	Camera Variance	0.9	0.7125	0.7025	0.6410	0.8014	0.8118	0.7917	0.7435
EX819	Wrap Chance	0.1	0.6716	0.6504	0.5918	0.8295	0.8428	0.8131	0.7332
EX820	Wrap Translucency	0.1	0.7274	0.7193	0.6693	0.8269	0.8436	0.8293	0.7693
EX821	Texture Opacity Intensity	0.1	0.6908	0.6686	0.6138	0.8117	0.8112	0.7693	0.7276
EX822	Texture Opacity Variance	0.9	0.7190	0.7177	0.6830	0.8260	0.8340	0.8157	0.7659
EX823	Texture Chance	0.9	0.7365	0.7144	0.6537	0.8281	0.8397	0.8297	0.7670

Tabulka 5.4: Tabulka výsledků 3. fáze. Testy probíhaly na hranicích 30%, 40% a 50%, v tabulce jsou pro lepší čitelnost sloučeny. Modrý řádek jsou výsledky modelu s výchozími hodnotami, zelená jsou modely které měly větší úspěšnost než referenční výchozí modely, zbylé experimenty měly nižší úspěšnost a byly tedy vyřazeny. Pro rozhodování jsem se řídil hlavně metrikou accuracy u obou typů přepravek. Hodnoty parametrů u více úspěšných datových sad se staly novými výchozími hodnotami, zbylé hodnoty zůstaly stejné.

fázi se dále prozkoumají hodnoty 0.25 a 0.35. Z těchto experimentů se nakonec vyberou ty nejlepší hodnoty, a vznikne konečná kombinace hodnot parametrů scény, které se použijí pro finální datovou sadu.

Jak lze vidět v tabulce 5.4, tak se ve třetí fázi nalezlo mnoho nových, více úspěšnější hodnot, i když rozdíly úspěšnosti nebyly tak výrazné jako v předchozí fázi. U celkem 7 parametrů scény se našla lepší hodnota. Výsledky třetí fáze jsou v tabulce 5.5.

Na rozdíl od třetí fáze, čtvrtá fáze byla méně úspěšná, jak lze vidět v tabulce 5.6. Z celkem 20 experimentů, neměla ani jediná datová sada větší úspěšnost než referenční. Každopádně ve většině případů nebyl rozdíl mezi referencí výrazný. Tento neúspěch je ale ve výsledku prospěšný, jelikož ukázal, že menší krokování by už nemělo smysl.

Nakonec po všech fázích se nalezly ideální hodnoty všech parametrů scény. Tyto hodnoty dále použiji pro vytvoření finálního plně syntetické datové sady. Hodnoty jsou v tabulce 5.7

Phase 4 default values

Light Intensity	0.3	Box Color Variance	0.9	Wrap Translucency	0.1
Light Variance	0.5	Camera Height	0.0	Texture Opacity Intensity	0.0
Daylight Intensity	0.9	Camera Variance	1.0	Texture Opacity Variance	0.9
Daylight Variance	0.0	Wrap Chance	0.2	Texture Chance	0.9

Tabulka 5.5: Seznam použitých výchozích hodnot parametrů pro fázi 4. Tyto hodnoty byly vybrány jako nejslibnější hodnoty parametrů v 3. fázi.

Dataset	Parameter	Value	F1-score						Average
			E1 30%	E1 40%	E1 50%	E2 30%	E2 40%	E2 50%	
EX824	Reference (default values)		0.7262	0.7268	0.6956	0.8222	0.8276	0.7940	0.7654
EX825	Light Intensity	0.25	0.6896	0.6706	0.6274	0.8537	0.8484	0.8191	0.7515
EX826	Light Intensity	0.35	0.6508	0.6408	0.5951	0.8346	0.8308	0.8021	0.7257
EX827	Light Variance	0.45	0.6978	0.6889	0.6348	0.8322	0.8404	0.8192	0.7522
EX828	Light Variance	0.55	0.6948	0.6598	0.5805	0.8197	0.8161	0.7910	0.7270
EX829	Daylight Intensity	0.85	0.6697	0.6777	0.6516	0.8191	0.8213	0.7903	0.7383
EX830	Daylight Intensity	0.95	0.7344	0.7360	0.7074	0.7841	0.7911	0.7636	0.7528
EX831	Daylight Variance	0.05	0.6229	0.6107	0.5632	0.8402	0.8345	0.8060	0.7129
EX832	Box Color Variance	0.85	0.6700	0.6618	0.6126	0.8557	0.8545	0.8266	0.7469
EX833	Box Color Variance	0.95	0.7025	0.6901	0.6287	0.8085	0.8158	0.7866	0.7387
EX834	Camera Height	0.05	0.7156	0.7105	0.6605	0.8182	0.8245	0.8032	0.7554
EX835	Camera Variance	0.95	0.6743	0.6648	0.6189	0.8239	0.8255	0.7970	0.7341
EX836	Wrap Chance	0.15	0.6360	0.6126	0.5554	0.8358	0.8325	0.7988	0.7118
EX837	Wrap Chance	0.25	0.6742	0.6606	0.6155	0.8360	0.8390	0.8092	0.7391
EX838	Wrap Translucency	0.05	0.6965	0.6813	0.6368	0.8269	0.8157	0.7795	0.7395
EX839	Wrap Translucency	0.15	0.6326	0.6204	0.5814	0.8343	0.8276	0.8057	0.7170
EX840	Texture Opacity Intensity	0.05	0.7003	0.6979	0.6663	0.8236	0.8403	0.8319	0.7601
EX841	Texture Opacity Variance	0.85	0.6219	0.6130	0.5758	0.8227	0.8145	0.7770	0.7042
EX842	Texture Opacity Variance	0.95	0.6750	0.6540	0.6015	0.7982	0.7931	0.7638	0.7142
EX843	Texture Chance	0.85	0.6480	0.6504	0.6289	0.8195	0.8250	0.7960	0.7280
EX844	Texture Chance	0.95	0.7044	0.6807	0.6023	0.8152	0.8216	0.7975	0.7370

Tabulka 5.6: Tabulka výsledků 4. fáze. Testy probíhaly na hranicích 30%, 40% a 50%, v tabulce jsou pro lepší čitelnost sloučeny. Modrý řádek jsou výsledky modelu s výchozími hodnotami. Nenašla se ani jedna lepší hodnota, což může značit, že krokování bylo již moc malé.

Final values of all scene parameters

Light Intensity	0.3	Box Color Variance	0.9	Wrap Translucency	0.1
Light Variance	0.5	Camera Height	0.0	Texture Opacity Intensity	0.0
Daylight Intensity	0.9	Camera Variance	1.0	Texture Opacity Variance	0.9
Daylight Variance	0.0	Wrap Chance	0.2	Texture Chance	0.9

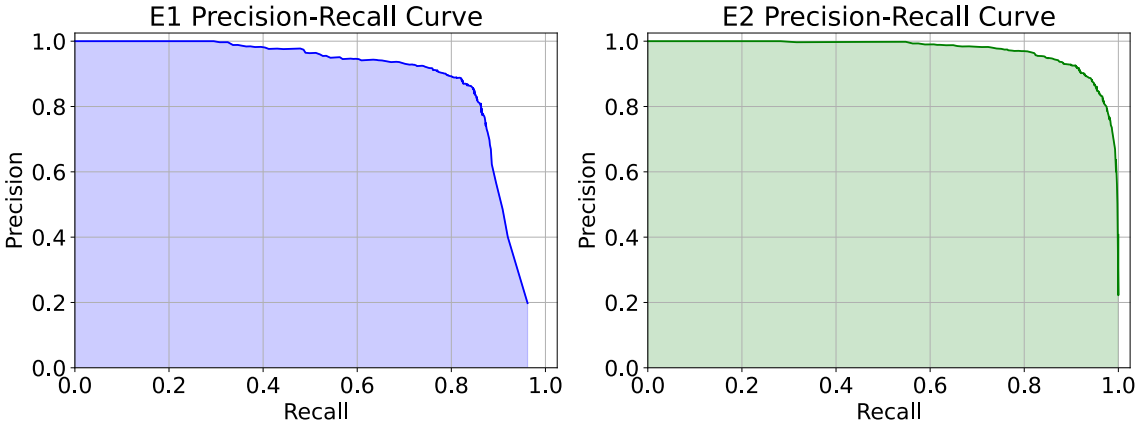
Tabulka 5.7: Seznam hodnot parametrů scény které prošly všemi fázemi. Jedná se o nejlepší nalezenou kombinaci parametrů scény, a pomocí této kombinace se vytvoří finální datová sada.

5.3 Výsledky samotných modelů

Jako první jsem vyzkoušel plně syntetickou datovou sadu s 20 000 snímku, z toho 16 000 trénovacích a 4 000 validačních snímků. Model měl při $IoU = 0.4$ F1-skóre 0.8425 a 0.9196

Class	E1			E2		
Threshold	30%	40%	50%	30%	40%	50%
Precision	0.86	0.89	0.9	0.89	0.9	0.92
Recall	0.84	0.81	0.79	0.94	0.92	0.91
F1-Score	0.85	0.85	0.84	0.91	0.91	0.92

Tabulka 5.8: Tabulky výsledků modelů z čistě syntetických snímků. Byly vyzkoušeny hranice 30%, 40% a 50%, jelikož cíl práce nevyžaduje vysokou hladinu jistoty.



Obrázek 5.14: Precision-recall křivka přepravek E1 a E2, z modelu natrénovaného na 10 000 syntetických snímcích na hladině $IoU = 0.4$.

pro individuální přepravky E1 a E2. Poté jsem udělal další model s pouze polovinou snímků, který měl téměř identické výsledky (tolerance ± 0.01). Na hladině $IoU = 0.5$ se rozdíly mezi modely neměnily, model s 20 000 snímky měl F1-skóre 0.7095 a 0.8740, model s méně snímky měl opět srovnatelné výsledky. Můj cíl je pouze natrénovat 2 třídy pomocí sofistikované CNN, proto při velkém množství snímků jsem narazil na tuhle klesající míru návratnosti. Každopádně tohle také znamená, že jsem narazil na strop, jak moc kvalitní model mé syntetické data mohou vytvořit (při tomhle úkolu), takže tohle pokládám za úspěch.

Jako model při výsledném vyhodnocování budu používat model vytvořený z 10 000 snímků, jelikož u problematické třídy E1 má vyšší F1-skóre o 0.0029, a mírně nižší skóre u neproblémové třídy E2. Jako referenční reálný model budu využívat model vytvořený z 160 reálných snímků. Je ale důležité upozornit, že jsem měl k dispozici jen pár videí, ze kterých jsem musel vytvořit jak testovací, tak i trénovací data. Proto mezera mezi trénovacími a testovacími snímky je velmi nízká. Aby jsem získal dostatek dat, tak jsem často musel z video segmentu přepravy jedné palety udělat 3 snímky. Tedy reálný model má výhodu v tom, že se jeho trénovací snímky velmi podobají testovací sadě, zatímco syntetický model má výhodu v počtu snímků.

Tyto data jsou v grafu 5.8. Testoval jsem na hranicích jistoty 30 %, 40 % a 50 %. Jak jsem očekával, tak i zde byl rozdíl mezi přepravkou E1 a E2, a F1-skóre bylo ve všech případech nad 0.84. Myslím si, že se jedná o poměrný úspěch, protože i přesto že jsem využil 10 000 snímků, tak se jednalo o čistě syntetické snímky.

V obrázku 5.14 lze vidět precision-recall křivku (PRC) tohoto modelu na hladině $IoU = 0.4$. Využíval jsem menší hodnotu, hlavně kvůli menším bounding box u testovací sady. Hodnota AP_40 (average precision na IoU=0.4) je 0.875 pro E1 a 0.973 pro E2, tedy mAP (mean average precision) pro model s těmito třídami je 0.924.

Na první pohled je vidět, že model je více spolehlivý u přepravek E2 – tento jev je i u referenčního reálného modelu. Tohle lze vysvětlit tím, že tvar přepravky E1 je podmnožinou E2, E2 má více identifikovatelný tvarů (5 otvorů, zatímco přepravka E1 má pouze 3) a zároveň se na paletách s E2 přepravkami nepoužívají fólie. Tyhle faktory značně usnadňují detekci přepravek E2.

5.4 Výsledky a porovnání metod při detekci na statických snímcích

V této kapitole budu porovnávat dva modely, reálný a syntetický, a u každého z nich budu dále porovnávat čistou detekci, a poté detekci s využitím metod pro následné úpravu (post-processing). Povedlo se mi implementovat tři metody, které jsem zmínil v kapitole 3.4. Dvě metody na nalezení možných objektů, které se nepovedly predikovat, a jedna metoda na odfiltrování potencionálních nepravdivých pozitiv.

Jelikož není žádná standardní metoda na vyhodnocení výsledků tohoto typu, tak jsem tohle musel testovat manuálně. Postup byl jednoduchý, pomocí programu 3.7 jsem si zvolil nastavení, které vypadalo nejlépe na ukázkových snímcích. Tohle nastavení jsem použil při testování všech snímků, aby to bylo férové. Použil jsem výchozí nastavení, kromě prahu detekce, které jsem nastavil na nízkou hodnotu 0.4, jelikož se i při této nízké hodnotě nepredikovalo velké množství nepravdivý pozitiv a už od dřívějších experimentech vím, že bude problém spíše s nedetekováním všech přepravek, než s vytvořením nesprávných predikcí. Následně jsem otestoval snímek po snímku a poznamenal jsem si podstatné metriky pro každý snímek, tedy:

- Skutečný počet přepravek ve snímku.
- Počet predikcí od detektoru.
- Počet predikcí správně odstraněných pomocí shlukovacího algoritmu.
- Počet predikcí nesprávně odstraněných pomocí shlukovacího algoritmu.
- Počet správně vytvořených predikcí.
- Celkový počet predikcí s využitím následného zpracování.
- Nesprávné predikce od detektoru.

Pro evaluaci využiji jednoduché vzorce:

$$accuracy = \frac{predicted - false}{expected} \tag{5.6}$$

$$error = \frac{false}{expected} \tag{5.7}$$

kde *predicted* je celkový počet predikcí s využitím následného zpracování (v případě jeho nevyužití, je to počet predikcí od detektoru), *false* je suma nesprávných predikcí od detektoru a nesprávně odstraněných predikcí pomocí shlukovacího algoritmu, a *expected* je skutečný počet přepravek ve snímku.

Model	Class	Post-process	Accuracy	Error
Synthetic	E1	No	0.9024	0.0142
		Yes	0.9268	0.0142
	E2	No	0.9923	0.0192
		Yes	0.9923	0.0192
Real	E1	No	0.9431	0.0000
		Yes	0.9654	0.0020
	E2	No	0.9962	0.0000
		Yes	0.9962	0.0000

Tabulka 5.9: Výsledky testů na statických snímcích. Všimněte si třídy E2 – modely byly dostatečně účinné, že post-processing zde neměl žádný efekt.

V testovacích snímcích je celkem 752 přepravek na detekování, z toho 492 pro problematickou třídu E1 a 260 pro třídu E2. Třída E2 byla bezproblémová, vůbec se u testů této třídy nevyužily metody na následné zpracování. Tedy predikce nevytvářela nepravdivé pozitiva, ani nevynechávala přepravky uprostřed sloupců. E1 byla horší ale zajímavější, jelikož šlo vidět efekt následného zpracování. Velikost chyby byla většinou stejná před i po použití, jelikož metoda jen zřídka odstranila správnou predikci. Výsledky lze vidět v tabulce 5.9.

S těmito výsledky jsem spokojený, můj model ze syntetických dat se dostatečně přiblížil k modelu z reálných dat. Dále následné zpracování zvýšilo úspěšnost u třídy E1 o více než 2 %, a to u reálného i syntetického modelů.

5.5 Výsledky na videozáznamu

Videozáznam jsem testoval opět manuálně. Byl zde ale problém v nedokonalých datech. Ve videu jsou zaměstnanci, kteří pomocí vozíku přepravují palety s přepravkami z nákladního auta do skladu, každopádně často se některé nejspodnější přepravky dostanou mimo záběr kamery. Nakonec jsem se rozhodl mít přísnější přístup a očekávat, že můj program bude detekovat i tyhle přepravky, které nejsou v záběru po celou dobu, kromě extrémních případů.

Ve mém testovacím videozáznamu pro testování přepravek E1 se přepravilo celkem 21 palet z nákladního auta do skladu. Využil jsem opět můj syntetický model a referenční reálný model, s využitím následného zpracování a bez pro porovnání. Program správně v obou případech vytvořil 21 skupin snímků, kdy jedna skupina reprezentuje jednu cestu palety z dodávky do skladu, každopádně tohle je ovlivněno nastavením programu. Znalý uživatel může měnit parametry v případech, že se špatně vytvářejí skupiny. Každá z těchto palet měla 34 přepravek, dohromady je tedy 714 přepravek, které je potřeba správně detekovat.

Při vytvoření skupiny snímků a odfiltrování snímků (vysvětlení v kapitole 3.5) se vypočítá její hodnota modus, tedy nejčastější hodnota. Tohle je výsledný počet přepravek, které se převezly z dodávky do skladu (nebo naopak, v případe záznamu s E2).

Ve videozáznamu s přepravkami E2 je celkem 11 skupin, a každá paleta veze 20 přepravek. Každopádně palety byly často mimo záběr kamery, nakonec jsem vyhodnotil, že je rozumné očekávat, aby se detekovalo celkem 204 přepravek. Nakonec se jich úspěšně detekovalo 185 až 194, dle modelu a podle toho, jestli se využilo následné zpracování. Tyto výsledky i výsledky s přepravkami E1 jsou v tabulce 5.10.

Class	Model	Post-process	Accuracy
E1	Synthetic	No	0.8978
		Yes	0.9496
	Real	No	0.9608
		Yes	0.9720
E2	Synthetic	No	0.9069
		Yes	0.9167
	Real	No	0.9510
		Yes	0.9510

Tabulka 5.10: Výsledky testů na videozáznamu. Porovnává se plně syntetický model s reálné, před a po využití post-processing. Všimněte si mírně horších hodnot než u testů ze statických snímků. Používaly se jiné testy a na jiném médiu, takže nějakou odchylku lze očekávat, každopádně tohle také značí obtížnější práci s videozáznamem, jelikož je zapotřebí správně vyfiltrovat snímky na začátku a na konci, aby se nezapočítávaly snímky, ve kterých ještě nejsou všechny přepravky v záběru kamery.

S výsledky jsem zcela spokojený. Povedlo se mi vytvořit generátor syntetických dat, který dokáže natrénovat model dostatečně na to, aby se velmi blízko přiblížil modelu, který byl trénován na reálných datech.

Kapitola 6

Závěr

Práce se především zaměřila na vytváření syntetické datové sady pomocí automatického procesu, ve kterém se pro každý snímek vytvořila různorodá scéna, přičemž mnohé její elementy byly ovlivněny náhodou i specifikovanými parametry. Skript využívá program Blender a jeho rozsáhlou podporu skriptování, a také vykreslovací engine Octane Render. Vypracoval jsem stovky experimentů na správné odladění, za účelem nalezení té ideální kombinace parametrů scény pro vytvoření co nejúspěšnější syntetické datové sady.

Výsledné snímky jsou fotorealistické a skript je automaticky anotuje. Jedná se o velmi robustní proces, který vytvoří jakoukoli velkou syntetickou datovou sadu, přičemž doba na vykreslení jednoho snímku při rozlišení 512×512 je pod 20 sekund i u starších PC. Syntetická data byly využity pro trénování modelů v síti CenterNet.

Cílem bylo počítat přepravky na kameře skladu, ale nechtěl jsem jen přímočaře detekovat počet přepravek a vypsat číslo, proto jsem se mírně odchýlil od původního plánu, a po získání predikcí je dále analyzuji. Na bázi statického snímku využívám tři metody, dvě na domyšlení chybějící predikce, a jednu na vyfiltrování možných nepravdivých pozitiv, tyto metody nazývám následné zpracování. Toto mělo značnou úspěšnost u méně úspěšných modelů, a v extrémních případech, například když detektor detekoval pouze 64 % přepravek, tyhle metody zvýšily úspěšnost až na 86 %. Každopádně můj výsledný model dosahuje úspěšnosti nad 90 %, a zde následné zpracování zvýší úspěšnost jen mírně, ale ne zanedbatelně.

Na bázi statických snímků jsem dosáhl úspěšnosti při počítání přepravek 94.74 % bez využití následného zpracování, a 95.96 % s jeho využitím, a to za pomoci modelu, který byl natrénován pouze mými syntetickými daty.

Nakonec jsem využil videozáznam. Související snímky seskupuji pomocí detekce směru pohybu predikcí a porovnáním s předešlými snímky. Pokud se změní směr, nebo se přepravky dostanou mimo záběr, tak se ukončí skupina snímků. Následně se z této skupiny vypočítá hodnota modus. Tato hodnota je finální počet přepravek.

Ve videozáznamu jsem dosáhl úspěšnosti bez následného zpracování 89.78 % pro přepravku E1 a 90.29 % pro přepravku E2. S využitím následného zpracování jsem dosáhl úspěšnosti 95.01 % pro přepravku E1 a 91.67 % pro přepravku E2.

Pro pokračování navrhuji práci zaměřenou na využití jiného software – například Unreal Engine či Unity, jelikož si myslím, že je značný potenciál ve vytváření syntetických dat v reálném čase. Také by bylo možno prozkoumat jiné způsoby učení, konkrétně kombinaci syntetických a reálných dat pomocí přenosu učení či metody fine-tuning.

Literatura

[1] Barde, M. P. a Barde, P. J. What to use to express the variability of data: Standard deviation or standard error of mean? *Perspectives in Clinical Research* online. India: Indian Society for Clinical Research, Červenec 2012, sv. 3, č. 3, s. 113–116. Dostupné z: https://doi.org/10.4103/2229-3485.100662. [cit. 2024-03-29].

[2] Block, L.; Raiser, A.; Schön, L.; Braun, F. a Riedel, O. Image-Bot: Generating Synthetic Object Detection Datasets for Small and Medium-Sized Manufacturing Companies. *Procedia CIRP* online. Science Direct, Květen 2022, sv. 107, s. 434–439. ISSN 2212-8271. Dostupné z: https://doi.org/https://doi.org/10.1016/j.procir.2022.05.004. [cit. 2024-04-22].

[3] Burkov, A. *The Hundred-Page Machine Learning Book*. Andriy Burkov, leden 2019. ISBN 9781999579517.

[4] Buslaev, A.; Iglovikov, V. I.; Khvedchenya, E.; Parinov, A.; Druzhinin, M. et al. Albumentations: Fast and Flexible Image Augmentations. *Information*. MDPI AG, Únor 2020, sv. 11, č. 2, s. 125. ISSN 2078-2489. Dostupné z: http://dx.doi.org/10.3390/info11020125.

[5] Dumoulin, V. a Visin, F. A guide to convolution arithmetic for deep learning. online. 2, Srpen 2016. revidováno 11. 1. 2018. Dostupné z: https://doi.org/https://doi.org/10.48550/arXiv.1603.07285. [cit. 2024-04-18].

[6] Fawcett, T. An introduction to ROC analysis. *Pattern Recognition Letters* online. Science Direct, Leden 2006, sv. 27, č. 8, s. 861–874. ISSN 0167-8655. Dostupné z: https://doi.org/10.1016/j.patrec.2005.10.010. [cit. 2024-04-12].

[7] Goodfellow, I.; Bengio, Y. a Courville, A. *Deep Learning* online. 1. vyd. MIT Press, listopad 2016. Adaptive Computation and Machine Learning. ISBN 9780262035613. Dostupné z: http://www.deeplearningbook.org. [cit. 2024-04-07].

[8] Goodfellow, I. J.; Pouget Abadie, J.; Mirza, M.; Xu, B.; Warde Farley, D. et al. Generative Adversarial Networks. online. 1, Červen 2014. Dostupné z: https://doi.org/https://doi.org/10.48550/arXiv.1406.2661. [cit. 2024-04-12].

[9] Greff, K.; Belletti, F.; Beyer, L.; Doersch, C.; Du, Y. et al. *Kubric: a scalable dataset generator* online. Google Research, 2022. Dostupné z: https://github.com/google-research/kubric. [cit. 2024-04-14].

[10] Pailla, D. R.; Kollerathu, V. a Chennamsetty, S. S. Object detection on aerial imagery using CenterNet. online. 1, Srpen 2019. Dostupné z: https://doi.org/https://doi.org/10.48550/arXiv.1908.08244. [cit. 2024-04-22].

[11] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B. et al. Scikit-learn: Machine Learning in Python. online, 2011, sv. 12, s. 2825–2830. Dostupné z: https://scikit-learn.org/stable/about.html. [cit. 2024-04-14].

[12] Szeliski, R. *Computer vision: algorithms and applications* online. 2. vyd. Springer Nature, leden 2022. ISBN 3030343715. Dostupné z: https://szeliski.org/Book/. [cit. 2024-04-02].

[13] Tkachenko, M.; Malyuk, M.; Holmanyuk, A. a Liubimov, N. *Label Studio: Data labeling software* online. HumanSignal, 2020-2022. Dostupné z: https://github.com/heartexlabs/label-studio. [cit. 2024-04-14].

[14] Yu, F.; Wang, D.; Shelhamer, E. a Darrell, T. Deep Layer Aggregation. online. 3, Červenec 2019. revidováno 4. 1. 2019. Dostupné z: https://doi.org/https://doi.org/10.48550/arXiv.1707.06484. [cit. 2024-04-22].

[15] Zhou, X.; Wang, D. a Krähenbühl, P. *Objects as Points* online. 2019. Dostupné z: https://github.com/xingyizhou/CenterNet. [cit. 2024-04-14].

Přílohy

Příloha A

Obsah přiložených souborů

V této sekci je obsah přiložených souborů. Vynechal jsem některé méně důležité soubory, dokumentaci, či soubory, které jsem sám nevytvořil nebo neupravil, jako jsou například soubory od CenterNet.

Soubory jsou rozděleny do 3 částí – CenterNet, který má dvě funkce, a to trénování modelů a detekce objektů. Generátor syntetických dat je Blender API skript, který vytváří syntetické data. Jako poslední jsou dodatečné skripty. Tyto skripty pracují s datovými sadami, slouží k analýze výsledků, nebo jsem je využil pro vytváření grafů v této technické zprávě. Podrobné instrukce jsou v README souboru, který slouží jako manuál. Mimo tyto 3 části také přikládám Excel soubor s výsledky experimentů, jak jsem zmínil v kapitole 5.2.2.

CenterNet

```
centernet/
├── data/ ..... Place created datasets in here
│   └── Folders for individual datasets/
├── exp/ ..... When training, models will be saved here
│   └── ctdet/
│       └── Folders with trained models/
├── models/ ..... Stores final model files
│   └── Final model files
├── src/
│   ├── autotest.py ..... Script for testing
│   ├── image_detector.py ..... Run detection on static images
│   ├── video_detector.py ..... Run detection on video file
│   ├── train.py ..... Training models
│   ├── train_loop.sh ..... Easy-to-use bash script for training multiple models
│   └── train_once.sh ..... Easy-to-use bash script for training
└── tests/
    └── Folders with various test data/
```

Nejsem autorem projektu CenterNet. Základní kód jsem převzal, dále modifikoval a vytvořil navazující soubory pro mé potřeby.

Zdroj: github.com/xingyizhou/CenterNet

Generátor syntetické sady

```
creator/
├── data/
│   ├── hdri/ ..... Stores HDRI maps
│   │   └── HDRI files
│   ├── object/ ..... Stores all 3D meshes
│   │   ├── wrap_e1/
│   │   │   └── Wrap objects for E1
│   │   ├── wrap_2/
│   │   │   └── Wrap objects for E2
│   │   ├── box_e1.obj
│   │   └── box_e2.obj
│   ├── preview/ ..... Latest preview scene file and images for debugging
│   │   └── preview.blend
│   ├── texture/ ..... Stores all textures
│   │   └── Various textures
│   ├── scene.blend
│   └── default_config.json
├── output/ ..... Datasets will be created in here
│   └── Folders with created datasets/
└── src/
    ├── creator/ ..... Files for synthetic dataset generator
    │   ├── scene.py
    │   ├── creator.py ..... Main file of program
    │   ├── utility.py
    │   ├── bbox.py
    │   ├── constants.py ..... You can change advanced settings in here
    │   └── annotation.py
    ├── interface/ ..... Files for UI program
    │   ├── render.py
    │   ├── preview.py
    │   ├── conf_window.py
    │   ├── graphic.py
    │   ├── main_window.py
    │   └── popup.py
    ├── utils/ ..... Run this to install necessary packages for Blender
    │   └── install.py
    └── main.py ..... GUI program
└── batch_creator.py ..... Easier way to run the creator.py script
```

Funkce `redirectOutput()` v souboru `creator/src/creator/utility.py` je převzata, autorem je uživatel Marc Morcos.

Zdroj: blender.stackexchange.com/questions/6119/suppress-output-of-python-operators-bpy-ops

Pomocné skripty

```
other_scripts/
├── dataset_utils/ ..... Scripts for manipulating datasets
│   ├── split.py
│   ├── merge.py
│   └── reduce_size.py
├── generate_dataset_code/ ..... Creates files required by CenterNet
│   ├── generate_dataset_code.py
│   └── template.py
├── graph_maker/ ..... Creates graphs used in this document
│   ├── data/
│   ├── output_pdf/
│   ├── output_svg/
│   ├── make_all.py
│   ├── wrap_chance_render_time.py
│   ├── parameter_tests.py
│   ├── learning_rate_decay.py
│   ├── learning_rate.py
│   ├── precision_recall.py
│   └── svg_to_pdf.py
├── parse_results/ .....Parses output from detector and evaluates it
│   ├── ground_truth_V3/
│   ├── ground_truth_V4/
│   ├── ground_truth_V5/
│   └── main.py
└── wrap_maker/ ..... Creates 3D meshes of wraps to be used in generator
    ├── output/
    └── main.py
```

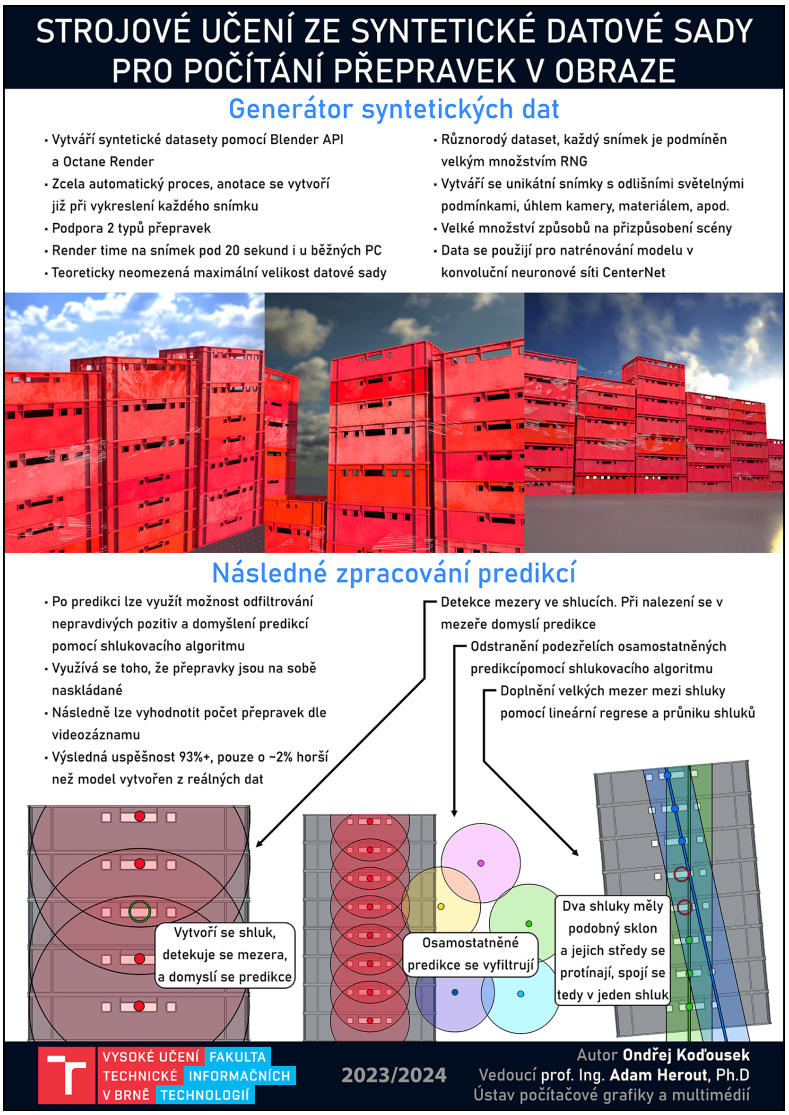
Funkce `calculateIntersectionUnion()` v souboru `other_scripts/parse_results/main.py` je převzata a upravena, autorem je Ahmed Gad.

Zdroj: kdnuggets.com/2021/03/evaluating-object-detection-models-using-mean-average-precision.html

Příloha B

Plakát

Přikládám i jednoduchý plakát k prezentaci mé práce.



Obrázek B.1: Plakát mé práce. Soubor s vyšší kvalitou je v přiložených souborech.