VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

GENERÁTOR SYNTETICKÝCH DAT PRO VÝVOJ ALGORITMŮ KOMPENZUJÍCÍCH ZKRESLENÍ VZNIKAJÍCÍ POHYBEM OBJEKTŮ SNÍMANÝCH ŘÁDKOVOU KAMEROU

SYNTHETIC DATA GENERATOR AIMED AT DEVELOPMENT OF ALGORITHMS FOR COMPENSATION OF DISTORTIONS CAUSED BY MOVEMENTS OF OBJECTS SCANNED BY A LINE SCAN CAMERA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Pavel Furik**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Pavel Škrabánek, Ph.D.**
SUPERVISOR

BRNO 2023

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Pavel Furik**
Studijní program:	Aplikovaná informatika a řízení
Studijní obor:	bez specializace
Vedoucí práce:	**Ing. Pavel Škrabánek, Ph.D.**
Akademický rok:	2022/23

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Generátor syntetických dat pro vývoj algoritmů kompenzujících zkreslení vznikající pohybem objektů snímaných řádkovou kamerou

Stručná charakteristika problematiky úkolu:

Zatímco klasické obrazové snímače dokážou zachytit obdélníkový výřez snímané scény jako celek v jeden jediný okamžik, řádkové kamery musejí pro získání stejné fotografie postupně snímat scénu řádek po řádku. To znamená, že řádková kamera nebo snímaný objekt se musejí pohybovat. V případě že při tomto postupném snímaní dojde k nechtěnému pohybu snímaného objektu nebo kamery, vznikne v obraze zkreslení. Toto zkreslení je nežádoucí a je třeba jej ve výsledném obraze co nejvíce potlačit. K vývoji takovéto metody je třeba dostatečně velká a reprezentativní sada anotovaných snímků. Takovéto sady však nejsou volně k dispozici, a jejich vytváření je časově i technicky náročné.

Cíle diplomové práce:

Student vypracuje rešerši problematiky generování syntetických dat pro úlohy počítačového vidění, a nastuduje problematiku řádkových kamer. Student analyzuje zadaný problém. Na základě analýzy pak navrhne a vytvoří software, který bude generovat oštítkované snímky určené k vývoji algoritmů kompenzujících zkreslení vznikající pohybem objektů snímaných řádkovou kamerou. Software bude uživateli umožňovat nastavení vlastnosti datové sady.

Seznam doporučené literatury:

MICHELUCCI, Umberto. Advanced Applied Deep Learning [online]. Berkeley, CA: Apress, 2019 [cit. 2020-09-10]. DOI: 10.1007/978-1-4842-4976-5. ISBN 978-1-4842-4975-8.

ROZANTSEV, Artem, Vincent LEPETIT a Pascal FUA. On rendering synthetic images for training an object detector. Computer Vision and Image Understanding [online]. 2015, 137, 24-37 [cit. 2020-09-10]. DOI: 10.1016/j.cviu.2014.12.006. ISSN 10773142. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S1077314214002446.

MILL, Leonid, David WOLFF, Nele GERRITS, et al. Synthetic Image Rendering Solves Annotation Problem in Deep Learning Nanoparticle Segmentation. Small Methods [online]. 2021, 5(7) [cit. 2022-10-21]. ISSN 2366-9608. Dostupné z: doi:10.1002/smtd.202100223.

BATCHELOR, Bruce G., ed. Machine Vision Handbook [online]. London: Springer London, 2012 [cit. 2022-09-05]. ISBN 978-1-84996-168-4. Dostupné z: doi:10.1007/978-1-84996-169-1.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2022/23

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jiří Hlinka, Ph.D. děkan fakulty

ABSTRAKT

Diplomová práce se zabývá problémem deformace snímků vzniklé v důsledku pohybu při snímání řádkovou kamerou. Algoritmy kompenzující dané zkreslení často vyžadují velké množství dat pro jejich správnou funkci. Datové sady jsou často nedostupné a jediná možnost je využití syntetických dat. S dnešní výpočetní technikou lze syntetizovat snímky blízké reálným. V práci jsou shrnuty klasické metody generování syntetických dat a některé aplikace neuronových sítí. Další částí je popis vytvořeného softwaru, který produkuje snímky podobné chybnému výstupu řádkové kamery.

ABSTRACT

Master's thesis deals with problem of image motion deformation as a result of image capture by line scan camera. Algoritms that compensate said deformation often need large pool of data to ensure their correct function. Datasets are often not available, thus the need for synthetic data generation. With current graphical hardware it is possible to synthesize images close to reality. In this thesis are summarized common methods for image syntheis and few aplications of neural networks. The next part summarizes software implementation that generates images simillar to deformed output of line scan camera.

KLÍČOVÁ SLOVA

záznam snímků, řádkové kamery, tréninková data, neuronové sítě, vykreslování syntetických snímků, generování textur, deformace snímků, generátor dat

KEYWORDS

image capture, line scan cameras, training data, neural networks, rendering synthetic images, texture generation, image deformation, data generator

ÚSTAV AUTOMATIZACE A INFORMATIKY

2023

BIBLIOGRAFICKÁ CITACE

FURIK, Pavel. *Generátor syntetických dat pro vývoj algoritmů kompenzujících zkreslení vznikající pohybem objektů snímaných řádkovou kamerou.* Brno, 2023. Dostupné také z: `https://www.vut.cz/studenti/zav-prace/detail/149561`. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky, Vedoucí práce: Ing. Pavel Škrabánek, Ph.D.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato diplomová práce je mým původním dílem, vypracoval jsem ji samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků.

V Brně dne 21. 5. 2023 .....

Pavel Furik

OBSAH

1 ÚVOD **13**
2 PRINCIP SNÍMÁNÍ KAMEROVÝMI SENZORY **15**
3 GENEROVÁNÍ SYNTETICKÝCH SNÍMKŮ **17**
3.1 Syntetické snímky a neuronové sítě 20
3.1.1 Vybrané typy neuronových sítí 21
3.1.2 Neurální vykreslování 25
4 ANALÝZA PROBLÉMU **29**
5 ŘEŠENÍ FUNKCÍ GENERÁTORU **33**
5.1 Simulace dýchání 33
5.2 Generace textur s velkým rozlišením 34
5.3 Texturové mapy 36
5.4 Model osvětlení 38
5.5 Realizace barevných vzorů na kůži 40
6 GENERÁTOR SYNTETICKÝCH DAT **43**
6.1 Modely ještěrek 44
6.2 Generování textur a masek 45
6.3 Parametry generátoru 49
6.3.1 Náhodné parametry 49
6.3.2 Uživatelský vstup 50
6.4 Animace a záznam dýchání 51
7 DISKUZE VÝSLEDKŮ **55**
8 ZÁVĚR **57**
SEZNAM POUŽITÉ LITERATURY **59**
SEZNAM ZKRATEK A SYMBOLŮ **65**
SEZNAM OBRÁZKŮ **69**
SEZNAM PŘÍLOH **71**
A Příklady variací výstupu **73**
B Odkaz na software **77**
C Reálné a syntetické snímky **79**

1 ÚVOD

Řádkové kamery oproti kamerám s plošnými snímači mohou poskytovat lepší rozlišení nebo rychlejší záznam snímku. K jejich správné funkci je třeba pohyb kamery nebo objektu, který definuje směr snímání. Pokud dojde během záznamu snímku (řádku) k pohybu mimo tento směr vznikne na snímku nechtěná deformace.

Tuto deformaci lze kompenzovat mnoha způsoby. Například pomocí neuronových sítí, které potřebují velký datový set k natrénování své funkce. Pro reálné aplikace není takové množství dat volně dostupné a generování nových snímků nemusí být prakticky realizovatelné. Uměle vytvořená data mohou tvořit náhradu, případně poskytovat krajní variace, které jsou těžko získatelné. Generátor syntetických snímků by měl takovou sadu poskytovat.

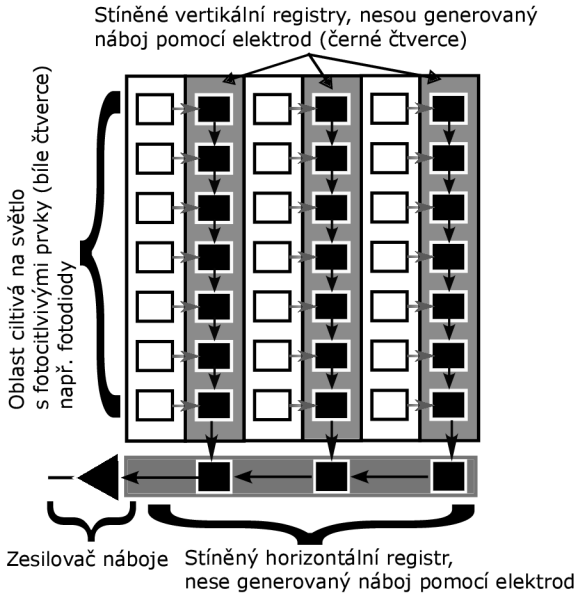
Generátor dat, jehož vytvoření je cílem této práce, má generovat data za účelem kompenzace deformace způsobené volným dýcháním ještěrek, snímané řádkovou kamerou. Toto dýchání není periodické a záleží na aktuálním psychickém a fyzickém rozpoložení ještěrky.

Teoretickou část práce tvoří stručné shrnutí problematiky snímání řádkovými kamerami. V této práci je dále uvažován pouze nechtěný pohyb objektu kolmý na směr snímání. Další části je rešerše metod generování syntetických dat pro aplikace v počítačovém vidění. Zde jsou uvedeny klasické metody generování snímků a některé vybrané aplikace neuronových sítí.

Z poznatků teoretické části a analýzy problému vychází software, který simuluje výstup řádkové kamery spolu s nechtěnou deformací a referenčním snímkem bez ní. Generátor by měl snímek označkovat a umožnit uživateli nastavení datové sady.

2 PRINCIP SNÍMÁNÍ KAMEROVÝMI SENZORY

Běžná kamera s plošným senzorem má obecně (M_y x N_x) (obr. 1), sloupců a řádků fotocitlivých prvků. Reálná scéna mapována pomocí optické soustavy na senzor, odpovídá rozměrům senzoru. Každá hodnota pixelu vychází ze záření, které dopadá na fotocitlivý prvek senzoru během expozice [5, 10].

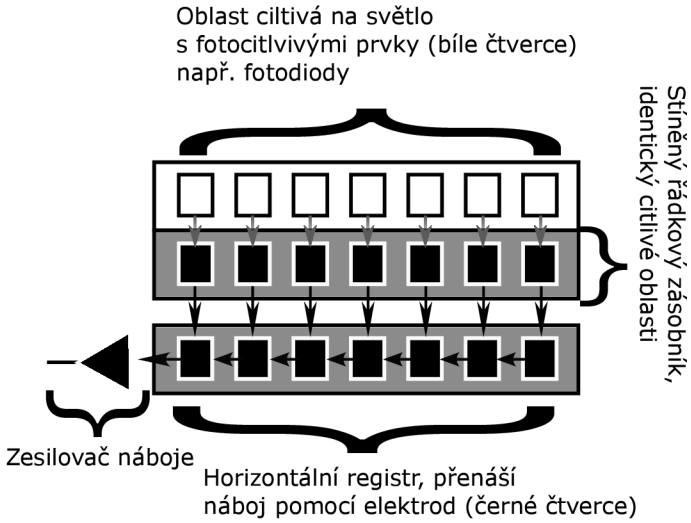


Obr. 1: Uspořádání plošného senzoru, Interline [5]

U řádkových kamer je tomu obdobně, ale senzor tvoří pouze jeden řádek fotocitlivých prvků (M_y x 1) (obr. 2). Zde M_y může být o stejné velikosti jako celkový počet prvků v plošném senzoru. K pořízení snímku je nutný pohyb kamery nebo objektu. Synchronizace tohoto pohybu se snímkovací frekvencí je nezanedbatelnou součástí kalibrace. Pokud kamera snímá řádky orientovány horizontálně, tak by pohyb měl nastávat pouze ve vertikálním ortogonálním směru. Používá se také označení cross (horizontální směr) a down (vertikální směr) nebo jednoduše směr snímání [5].

Jelikož pracujeme s dobou expozice okolo 0.1 až 1 ms/řádek, osvětlení musí být přizpůsobeno těmto hodnotám. Světlo je koncentrováno pouze na oblast snímaného řádku. Lze použít liniová světla [43], lasery a optická vlákna k vedení paprsků na požadované místo. Pro extrémně krátké expoziční časy je třeba zvážit zda intenzita světla není nevhodná pro snímaný objekt [5]. Řádková kamera může poskytovat mnohonásobně větší rozlišení v horizontálním směru s rychlejším záznamem oproti těm plošným [21]. Celistvý snímek vzniká spojením (stitching) jednotlivých řádků zaznamenaných kamerou, pomocí softwaru kamery. Počet řádků je roven po-

čtu snímků tvořící celistvý snímek. Omezení velikosti snímku ve směru snímání tvoří pouze dostupná paměť.



Obr. 2: Uspořádání řádkového senzoru s řádkovým zásobníkem [5]

Většina aplikací strojového vidění používá kamery s plošným senzorem. Vysokorychlostní aplikace, kde se objekt pohybuje ve scéně, vyžadují speciální osvětlení a kameru s dostatečně krátkou dobou záznamu. Zároveň nastává problém při snímání objektů větších než senzor kamery nebo kdy je celý objekt zaznamenám několika snímky, z důvodu jeho posunu. V obou případech je řádková kamera vhodná alternativa [5, 10].

Typickým použitím v oblasti strojového vidění je snímání nekonečných materiálových pásů. Příkladem může být výrobní linka s běžícím pásem, produkce papíru nebo textilních látek. Při použití kamery s plošným senzorem by bylo nutné kompenzovat část snímku, která se objeví na obou po sobě jdoucích snímcích. U řádkové kamery tento problém nenastane. Pokud by šířka objektu byla větší než co nám umožňuje snímat řádkový senzor, lze kamery řadit vedle sebe a kompenzovat případný konflikt v jednom řádku [5, 10].

Další typickou aplikací je snímání rotujících objektů s vyšším rozlišením. U válcových tvarů lze snímat například etikety plechovek, které se na snímku jeví v jedné rovině. Plechovka se jeví jako rozvinutá. Podobně u kruhových objektů snímaných na rotujícím podstavci, lze rozvinout objekt do jedné roviny řádků s vysokým rozlišením. Transformace souřadnic obnoví původní kruhový tvar [5].

Je nutné podotknout, že dosavadní rozdělení řádkových a plošných senzorů se vztahuje k CCD senzorům. U CMOS je toto rozdělení spíše otázkou řízení. Každý pixel lze číst samostatně, což umožňuje plošnému senzoru fungovat jako u řádkové kamery [5, 10].

3 GENEROVÁNÍ SYNTETICKÝCH SNÍMKŮ

Aplikací strojového učení je v dnešní době nespočet a stále jich přibývá. V oblasti strojového vidění je situace obdobná a dochází zde k vzájemnému prolínání. Příkladem mohou být detektory přítomnosti objektů [24, 26, 33] a klasifikátory [4, 18, 38, 40, 49], které ke své funkci používají neuronové sítě převážně konvolučního typu. Ty ke své správné činnosti potřebují obrovskou sadu dat a obecně platí, že čím víc dat máme k dispozici tím lepší mohou být potencionální výsledky.

Data jsou pro některé aplikace volně dostupné. Existuje mnoho databází a datasetů, ze kterých lze čerpat. U všech aplikací tomu tak ale není. Z důvodu proveditelnosti, finančních nákladů a dalších faktorů nelze získat data (snímky), které jsou dostupné v dostatečném množství, variabilitě a relevantnosti k dané aplikaci [33].

Dnes je možnost využití syntetických snímků dobře známa a někdy i jediná varianta k získání dat [33]. Datový set se typicky rozděluje na trénovací, validační a testovací části, které by se neměli překrývat [22]. Nároky na velikost datového setu se tak zvyšují. Hlavní důvody, proč použít generátor syntetických snímků tvoří nedostatek reálných předloh nebo zdroj s nedostatečnou variací.

Označování snímků

Mnoho aplikací strojového vidění a učení vyžaduje správné označení (label) [22, 40, 33], charakteristické pro učení s učitelem. Jedná se o skutečnou hodnotu nebo objektivní pravdu, kterou se snažíme predikovat pomocí algoritmů. Označování dat z reálného světa se většinou musí provézt manuálně, a to pomocí smyslů člověka. Příkladem může být označení snímku, na kterém se nachází zvíře, právě tím o jaké zvíře se jedná.

Kromě uvedení typu zvířete se pro správnou funkci klasifikátoru musí uvézt i hranice, například obdélník, která zvíře těsně vyznačuje. Proces určování hranice objektu automatizují algoritmy strojového vidění, vyznačováním kontur a hran okolo zvířete. U čistě syntetických snímků je označování již součástí generátoru a mělo by být zcela automatizováno. Odstraní se tak jedno z úzkých míst tréninku [33], avšak pro některé aplikace nemusí být označení přítomna [26], typicky se jedná o učení bez učitele.

Rozdíl domén

Rozdíl domén je hlavní vlastnost oddělující syntetické a skutečné snímky. Vzniká při přechodu mezi virtuálním světem, na kterém je aplikace natrénovaná, a tím reálným, kde natrénované algoritmy už nemusí fungovat zcela správně. V realitě dochází k ovlivnění snímku kamerou, prostředím, pohybem a dalšími faktory. Syntetické data mohou být ochuzeny o artefakty, rozmazání pohybem, šumem a další ovlivnění přítomné v reálném snímku [40].

Metody dostupné k odstranění tohoto rozdílu jsou závislé opět na aplikaci. U detektorů přítomnosti [40] je nejdůležitějším faktorem variabilita, tedy množství objektů v různých pozicích, rotacích atd. Tyto variace u syntetických snímků zaručují funkci i v reálném světě, snižují tak rozdíl domén. U jiných aplikací [24] hraje hlavní roli realistické vykreslení scény, které slouží k snížení rozdílu domén..

Poměr snímků z domén

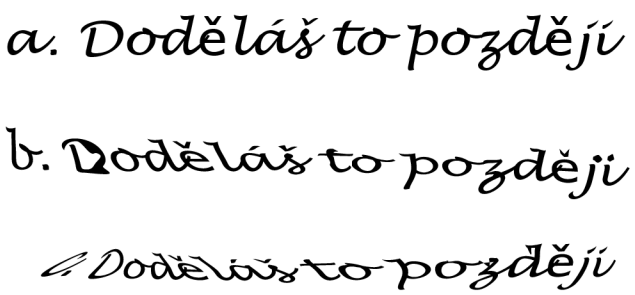
Konkrétní aplikace určuje vhodný poměr reálných snímků k syntetickým, například [18] používá pouze syntetické snímky, ale část klasifikační sítě byla již předem trénovaná na reálných datech. Příklady [40, 33] používají část snímků v různých poměrech a dosahují podobných výsledků. Další rozdělení syntetických dat, je na více či méně realistické snímky. Například [24] používá pouze fotorealistická data, zatímco [48] se zaměřuje na řízení robota a rychlost generace s jednoduchou scénou a tvary.

Pro danou aplikaci je třeba rozhodnout o poměru reálných a syntetických snímků. Zároveň snímky patřící do syntetické části mohou být rozděleny dle fotorealističnosti a dále určit poměr mezi nimi. Příliš velké množství syntetických snímků se negativně projeví na rozdílu domén [48].

Typy generování snímků

Zde jsou uvedeny vybrané metody generování syntetických snímků, u kterých variace nemusel manuálně vytvářet člověk kreslením na papír nebo digitálně. K generování syntetických snímků lze postupovat několika způsoby, které více či méně vycházejí z reálných dat.

Nejjednodušší metodou, je použití kopie reálného snímku a aplikace jednoduché transformační funkce. Metoda tedy počítá s existencí reálných snímků. Transformace produkují relevantní variace, které se zařadí do datového setu. Příkladem může být psaný text (obr. 3).

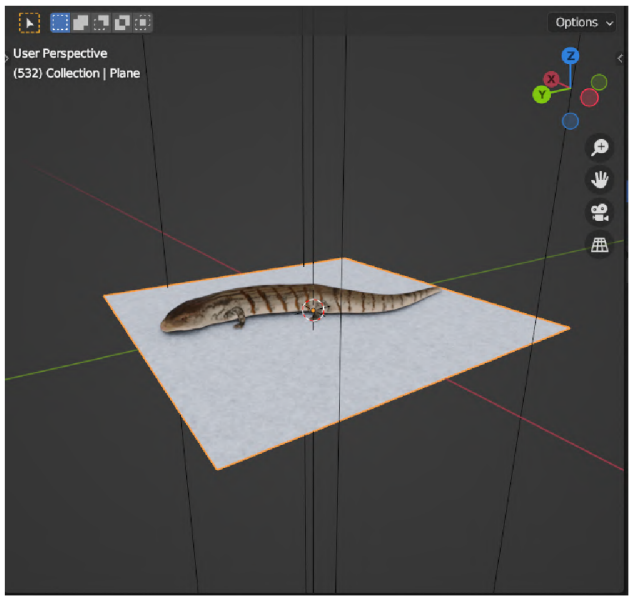


Obr. 3: Transformace textu ze zdroje s malou variací [49]

Další možností jsou kompozice reálných a fiktivních scén. Ze snímku se extrahuje pouze objekt zájmu a překryje se jím vygenerované pozadí. Výhodou tohoto postupu je jednoduchost a práce s 2D daty, příkladem může být dosazení textu

do scén a jeho detekce [17]. Nevýhodou je, že variace v rotaci a osvětlení je u 2D obrázku těžko proveditelná.

Opačný postup je také relevantní [40]. Reálný objekt je ze snímku odstraněn a na jeho místo dosazen syntetický model. Pokud má generátor vytvářet variace objektu, pak je tato kombinace vhodnější než předešlá, jelikož 3D model umožňuje snadnější variace natočení, osvětlení, postprocessing atd.



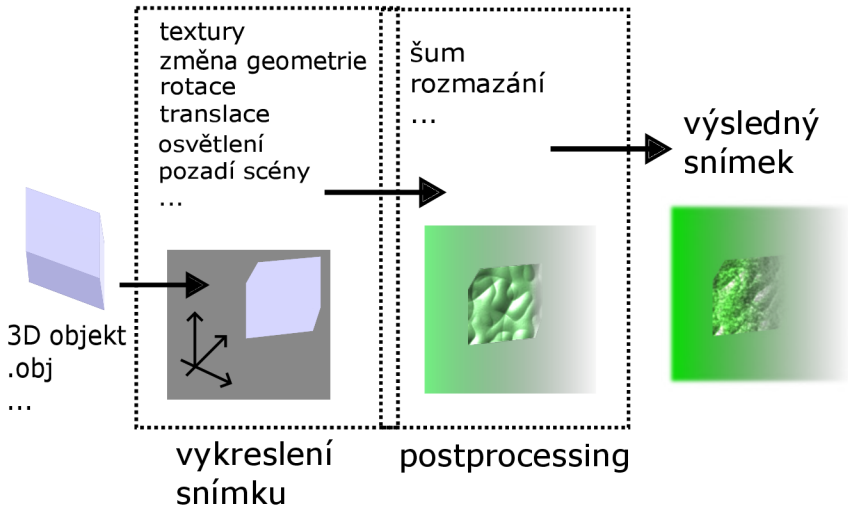
Obr. 4: Prostředí v Blenderu [15]

Model v 3D lze také doplnit fiktivním pozadím nebo celou scénu vykreslovat pomocí 3D softwaru, jako je Blender [33, 38], OpenGL [18], 3ds Max [24], případně herní engine [30] a mnohé další. Zde má uživatel kontrolu nad každým parametrem scény. Obvyklé změny parametrů, které slouží pro zvýšení variace scény jsou: Translace, rotace, výměna textur a změny geometrie s osvětlením (obr. 5). Variace parametrů působí pozitivně v určitém rozsahu, hodnoty mimo něj negativně ovlivní rozdíl domén. Výhodou 3D vykreslení mohou být fotorealistické snímky a kontrola nad parametry [24], ale za cenu vyšších výpočetních nároků [33].

Postprocessing

Syntetické snímky nikdy nebudou bezchybně reprezentovat realitu. Rozdíl domén nelze zcela odstranit, ale pomocí variace parametrů generátoru by se syntetické snímky měli přibližovat realitě. Syntetický snímek je ochuzen o některé vlastnosti reálných snímků (viz. rozdíl domén v kapitole 3), proto je nutný postprocessing.

Tato etapa zahrnuje operace, navazující na vykreslení snímku, ale také během něho. Etapy se tedy mohou překrývat (obr. 5). Patří zde úprava kontrastu, barev, šum, filtrování a další. Operace, které v postprocessingu reálných snímků slouží k odstranění nechtěných vlastností, mají invertovanou funkci, kde naopak syntetický sní-



Obr. 5: Obecná posloupnost operací při vykreslování [18]

mek negativními vlastnostmi obohacujeme. Při ablaci jednotlivých částí této etapy generátoru, vychází najevo důležitost přidaného šumu a rozmazání snímku k překonání rozdílu domén [18]. Obvykle se přidává náhodný a aditivní (Gaussovský) šum.

$$\frac{1}{256} \cdot \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Obr. 6: Příklad masky 5×5 aproximující Gaussovské rozložení.

Při specifických vlastnostech kamery je postprocessing vhodný způsob jak simulovat její efekt na snímek nebo video. Například aplikace vykreslování v reálném čase [25], simuluje starší kameru kde dochází zkreslení v důsledku její kvality, barevné filtry, čočky. Scéna vytvořena v OpenGL s vybranými kroky postprocessingu přidává artefakty a zkreslení snižující rozdíl domén mezi vytvořeným snímkem a tím skutečným.

3.1 Syntetické snímky a neuronové sítě

Jak už bylo naznačeno kapitolou 3, v poslední době dochází k velkému rozvoji neuronových sítí, obzvláště v oblasti generování syntetických obrázků. Implementace se stávají zastaralými a nemoderními stále rychleji, někdy i ve chvíli jejich implemen-

tace. Z tohoto důvodu je tato sekce omezena pouze na některé základní aplikace a typy neuronových sítí relevantní pro generátor syntetických snímků.

I tak je kombinací v jakých lze dosadit neuronovou síť do (obr. 5) velice mnoho. Zasadit ji lze před i za každou část a pokusit se tak snížit rozdíl domén [39], nebo jednotlivé části zcela nahradit neuronovou sítí. Extrém v tomto ohledu je neuronové vykreslování, kdy celou posloupnost (obr. 5) nahradíme neuronovou sítí.

3.1.1 Vybrané typy neuronových sítí

Generativní modely produkují nové data dle naučené vzájemné závislosti dat. Příkladem může být obraz X a jeho pixely. Ty mají mezi sebou skrytou závislost, která jim dovoluje něco reprezentovat, například psa, kočku, auto, obličej člověka atd. Síť je schopna naučit se rozdělení pixelů v obrazu, $P(X)$, které reprezentuje psa, číslo, písmeno atd. Cílem je ale vytvořit zcela novou variaci obrazu, tedy čerpat neomezené množství nových kombinací obrazů, které byly předloženy během tréninku.

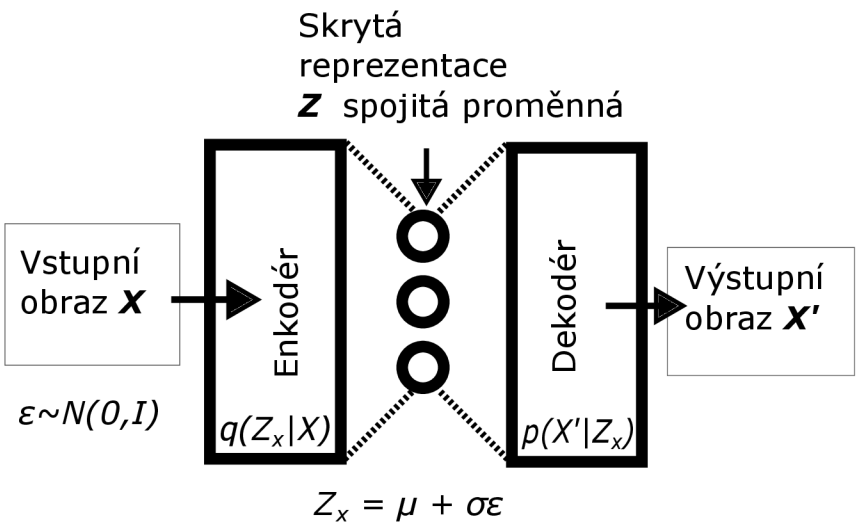
Generovaný snímek by měl být více fotorealistický, jelikož strukturu reálného obrazu lze těžko vyjádřit standardním vykreslením pomocí ručně nastavených parametrů generátoru nebo člověkem. Hlavní myšlenka je nechat neuronovou síť naučit se reálnou strukturu a získat snímek bližší realitě [12].

Variační autoenkodér VAE

Síť enkodér q_θ a dekodér p_θ tvoří dvě části VAE, čímž se podobají standardnímu autoenkodéru (obr. 7). Síť enkodér se snaží X redukovat reprezentací s co nejmenší velikostí v skrytém prostoru Z. Síť dekodér dostane na vstupu tuto reprezentaci, Z_x. Spojitou proměnou se snaží p_θ opět reprezentovat v prostoru původního X. Mezi nimi existuje latentní prostor skrytých proměnných. Ty jsou podstatně menší než X (obraz) a tvoří tak úzké místo, které reprezentuje pouze podstatné vlastnosti obrazu [12].

Rozdíl oproti standardnímu autoenkodéru tvoří spojité skryté proměnné. Zde je hlavní myšlenka taková, že skrytá proměnná Z_x má normální (Gaussovské) rozdělení $N(0, I)$, kde I je jednotková matice. Pixely předloženého snímku X, reprezentující psa v Z_x, normální rozdělení určitě mít nebudou, každopádně existuje funkce, která mapuje Z_x do rozdělení reprezentující psa. Jelikož neuronové sítě mohou sloužit jako aproximátor funkcí, nezbývá než tuto funkci odhadnout [12].

Díky triku náhodného výběru ε_N v prvních etapách výpočtů lze Z_x reprezentovat pomocí střední hodnoty μ a směrodatné odchylky σ spojitě. Chybu lze tak šířit pomocí backpropagation. Nevýhodu VAE může být rozmazání výsledného obrazu X' [12, 16].



Obr. 7: Obecná struktura VAE [12]

Generativní nepřátelské sítě GAN

Další generativní zástupce jsou sítě typu GAN (obr. 8). Skládají ze dvou sítí, ale tentokrát fungují proti sobě. Síť typu generátor produkuje snímky $G(X)$. Zatímco síť typu diskriminátor funguje jako klasifikátor, který se snaží určit jestli je předložený snímek reálný X nebo syntetický $G(X)$. V průběhu tréninku by tyto sítě měly dosáhnout rovnováhy, kdy diskriminátor nedokáže rozeznat syntetický snímek od reálného [16].

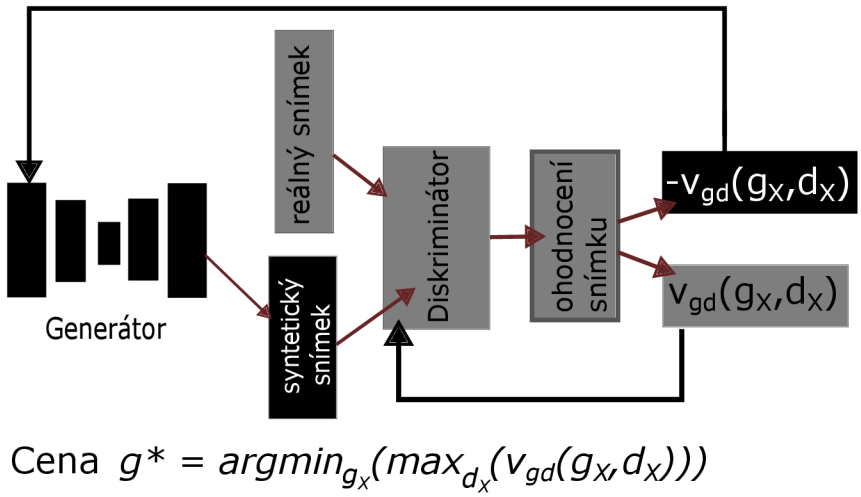
GAN byly původně určeny k aplikacím strojového vidění, často používají hluboké konvoluční neuronové sítě, například DCGAN [16]. Původní koncepce GAN je chápána jako hra s nulovým součtem. Generátor g_X se snaží minimalizovat získané ocenění diskriminátoru $argmin$, zatímco diskriminátor d_X se snaží získat maximum z ocenění $v_{gd}(g_X, d_X)$. Hodnota ohodnocení se vrací pomocí backpropagation k vahám obou sítí (obr. 8).

Natrénovaná GAN produkuje fotorealistické snímky. Nevýhodou může být nedosažení rovnováhy generátoru a diskriminátoru. Hodnoty gradientu s nekonečně malými hodnotami náhodně krouží kolem asymptoty, ale nikdy se k ní nedostanou [16].

Difuzní modely

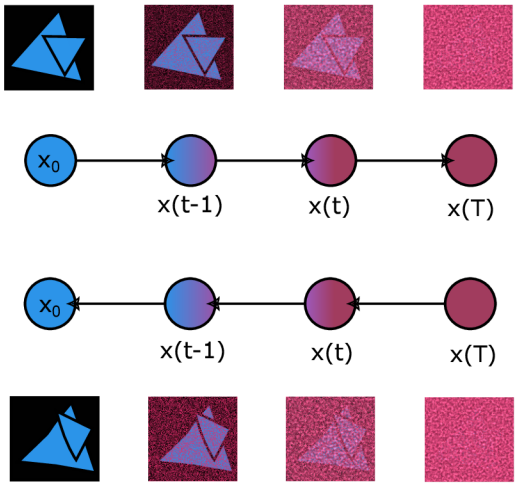
Difuzní modely mohou sloužit jako alternativa k GAN, které sice poskytují fotorealistické snímky, ale existuje problém jejich rozšíření do jiných domén, na nové typy snímků. Difuzní modely lze v tomto ohledu snadněji trénovat s větší diversitou [11].

Rozdíl oproti GAN spočívá v iterativním odstraňování šumu. GAN model dostane šum jako vstup a snaží se ho odstranit v jednom kroku, zatímco difuzní modely tento proces rozdělují na mezikroky různých velikostí (obr. 9). Počáteční



Obr. 8: Obecná struktura GAN [44]

obraz není zašuměný. Dle difuzního modelu a jeho specifického plánu dochází k postupnému přidání šumu. Proces zašumění a odstranění šumu jsou chápány jako dva Markovské řetězce, které jdou v opačných směrech [19].



Obr. 9: Příklad řetězců difuze [19]

Pro získání původního obrazu se síť naučí odhad šumu přítomného ve snímku. Tento odhad je odstraněn a následně znovu přidán v menším množstvím. Po T krocích dle (obr. 9) získáme podobný snímek. Je třeba uvézt, že kroky na (obr. 9) nemusí být v pořadí zde uvedeném. Kroky se v čase t mohou opakovat, případně přeskakovat k zvýšení rychlosti. Uplatňuje se zde znovu reparametrizační trik. Ten dovoluje kalkulovat přidaný šum pro krok v t předem místo postupného sčítání přes všechny kroky od x_0 do času t.

Kromě výše popsané části (obr. 9) obsahují difuzní modely často jazykový model, kterému člověk může dát podnět. Ten může mít podobu věty nebo jednot-

livých slov a určuje, jakým směrem se generace obrazu má dát (obr. 10). Většina dnes populárních difuzních modelů určených k produkci obrazů má vlastní seznam slovních podnětů [45].

Obr. 10: Výsledek difuze, podnět: Common lizard viewed from above.

Difuzní modely jsou schopny naučit se generovat snímky z mnoha domén. Pro tvorbu obrazů a napodobenin uměleckých děl, na kterých byly tyto modely natrénovány, je diverzita žádoucí. Problém může nastat u aplikací vyžadující přesné a stabilní generování snímků, například ve zdravotnictví [34, 3].

Často se používá trik řízení bez klasifikátoru, čímž se sníží variace výstupu modelu. Během tréninku se náhodně zamění slovní podnět za prázdný. Trénují se tak zároveň dva modely s podnětem a bez něho. Při generování snímku obě sítě produkují snímek a kombinují se. Experimentálně bylo zjištěno, že tento způsob produkuje snímky více relevantnější k slovnímu podnětu [20].

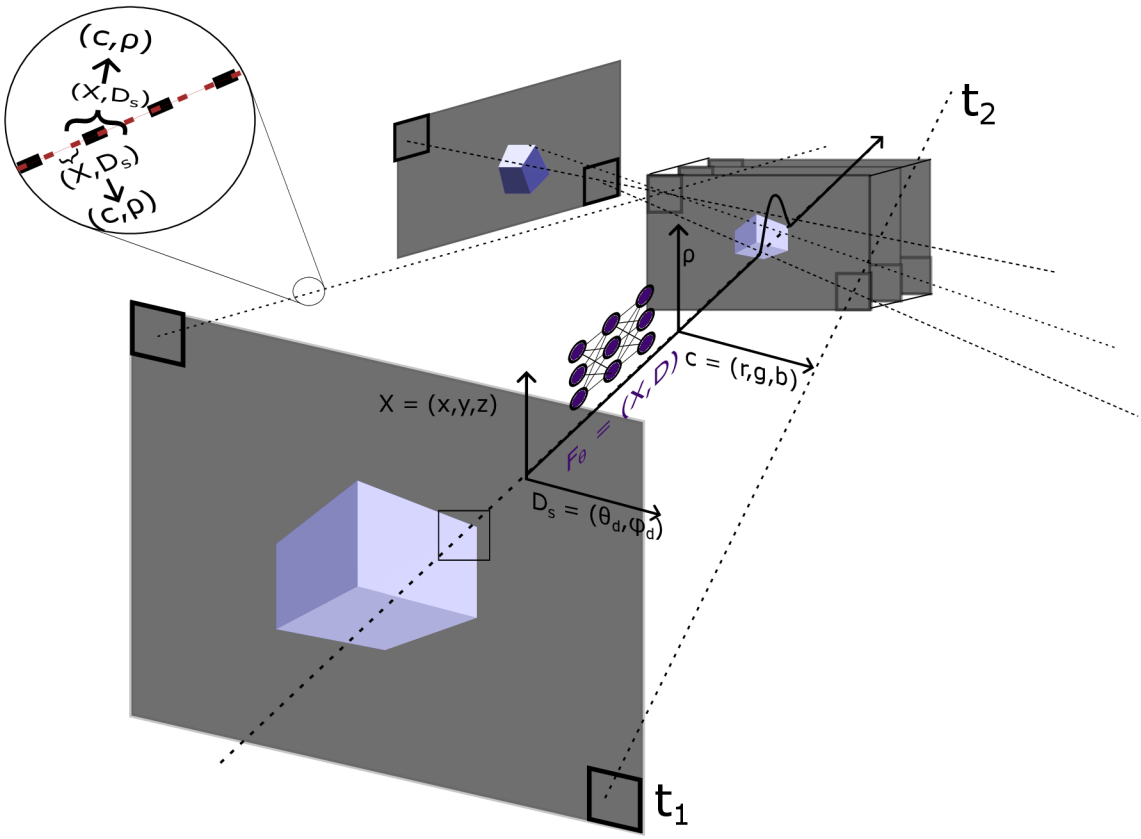
Neurální pole záře NeRF

Pro generaci 3D modelů nebo nových úhlů scény lze využít NeRF model. Vyznačují se přetrénováním na specifickou scénu nebo model, jehož strukturu chceme reprezentovat. K natrénování je třeba mnoho snímků z několika úhlů pohledu na objekt [31].

Vstup modelu tvoří dva vektory, lokace $X_{x,y,z}$ se souřadnicemi v 3D a směr paprsku s úhly $D_s = (\theta_d, \phi_d)$. Neuronová síť dle $F_\Theta : (X_{x,y,z}, D_s) \rightarrow (c, \rho)$ realizuje transformaci. Výstupem je RGB hodnota a hustota. Tato posloupnost operací se provede pro každý bod paprsku od t_1 do t_2 dvakrát. Druhý paprsek je vyslán (červeně obr.11) s jemnějším krokováním.

Informace získané z vyslání prvního paprsku, trénují první síť hrubě vykreslující scénu. Informace z jemnějšího paprsku trénuje druhou síť, která vykresluje scénu detailněji. Pomocí pole záře popsaného v [32] dojde k vykreslení nového pohledu na objekt ve scéně (obr. 11).

Hlavní výhodou těchto modelů tvoří struktura, kterou se NeRF naučí. Ta může být mnohonásobně kompaktnější něž tradiční popis scény pomocí modelu, textur, voxelů atd. Nevýhodu tohoto přístupu je přetrénování k reprezentaci konkrétní scény. Aby NeRF model reprezentoval jinou scénu, je třeba ho kompletně



Obr. 11: NeRF Raymarching [32]

přetrénovat. K reprezentaci třech různých scén jsou třeba tři natrénované NeRF modely [32].

3.1.2 Neurální vykreslování

Jak bylo uvedeno na začátku této kapitoly, čím dál častěji se do tradičního vykreslování (obr.5) dosazují modely neuronových sítí. Současná definice z [46] zahrnuje oblast vytváření snímků a videí pomocí hlubokého učení, umožňující měnit vlastnosti generovaných scén: osvětlení, parametry kamery, pozice, geometrie, vzhled a sémantickou strukturu. V [46] jsou uvedeny jako hlavní aplikace: Sémantická syntéza fotek, vykreslení nových úhlů pohledu, změna osvětlení a snímání pohybu obličeje a těla spolu s jeho rekonstrukcí.

Realistické vykreslování

Generativní neuronové sítě 3.1.1 dosahují často kvality, u které nelze pouhým okem rozlišit reálný snímek od syntetického. Vzhledem k dostupnosti obsáhlých databází, se toto týká obzvláště snímků lidských tváří nebo některé vybrané oblasti, oči, ústa atd. Jedna z nejdůležitějších vlastností generativních modelů, která jim umožňuje fungovat jako generátor specifických snímků, je rozšířením základních modelů o podněty a podmínky, které omezují náhodnost generovaných snímků [46].

Moduly počítačové grafiky

Klasické vykreslování lze kombinovat s neurálním, [46] označuje tento přístup jako modul počítačové grafiky. Příklad takového přístupu je [42], používá jako podmíněný vstup syntetický snímek vykreslený klasickým způsobem. Tento krok GAN (obr.8) přidává nový vstup generátoru a diskriminátor je jako vždy po tréninku odpojen. Jedná se o případ nasazení neuronové sítě pro vylepšení již existujícího syntetického snímku, nikoliv kompletně nového vykreslení.

Pro generování nových textur lze opět využít generativních modelů GAN [23, 6]. Natrénovaný model [23] dokáže generovat textury v nastavitelném rozlišení. Lze kombinovat několik podnětů dohromady k vytváření kombinovaných textur. Vygenerované textury jsou na sebe navazující (bezešvé), což umožňuje skládání do větších textur. Model [23] má problém generovat pravidelné ostré tvary, jako jsou rovné texty nebo textura šachovnice.

Druhý model GAN [6] navazuje na předchozí a zaměřuje se na periodické textury. Problémem je opět adaptace více domén. GAN modely mají tendenci některé data z tréninkového setu občas ignorovat [6]. Používání texturových map lze spojit s modulem počítačové grafiky. Generativnímu modelu je zpřístupněna informace o dalších texturách (texturové mapy), difuzní (albedo), odrazivost, normálová, bump a další [46].

Řízení parametrů generátoru

Pří vykreslování 3D scén klasickými metodami máme kompletní kontrolu nad jejími parametry. Modely neurálního vykreslování tuto vlastnost nemají nebo je omezená. O rozdílu domén rozhodují parametry popisující scénu (obr. 21), které neuronová síť dokáže lépe nastavit než člověk. Uživatel může parametry ovlivnit pouze vlastnostmi, které jsou přítomné v snímcích předložených během tréninku. Příkladem může být model, který mapuje pohyb snímku obličeje videa na jiný snímek [46].

Tréninkový set musí obsahovat popis dat označením, aby mohl uživatel řídit parametry neurálního vykreslování. Přidaný popis stěžuje trénink modelu a tvoří úzké místo (viz. kapitola 3). Většinou je snadnější natrénovat několik modelů s různými sety dat pro implicitní kontrolu, než se snažit tyto parametry popisovat [46].

Variace výstupu

Jako u generativních modelů, vykreslování celých scén ovlivňuje náhodnost daného generátoru. Na (obr. 10) výstupu difuzního modelu, je výběr z několika možností. Podobný výstup má i neurální vykreslování scén. Jedná se o několik variací vykreslené scény (osvětlení, barva, průhlednost, pozice objektů), ze kterých si uživatel může vybírat. Zde nastává opět problém mezi diverzitou výstupu a rozdílem domén [46].

Oproti generaci snímků (obr. 10), máme u scén větší kontrolu nad parametry, které můžeme měnit, například VAE dovoluje pomocí sémantického mapování

ovládat vlastnosti povrchu, jako jsou odlesky nebo průhlednost. Uživatel má přístup k těmto parametrům prostřednictvím uživatelského rozhraní [46].

Neurální vykreslování má problémy s větším množstvím domén. Většinou se přetrénují na specifickou scénu. Model který by v jedné scéně vykresloval psi jednoho plemene lze natrénovat. Výstupem by byl například kokršpaněl všech barev, variací textur a úhlů pohledu. Stejný model ale nedokáže vykreslit auta nebo lodě. Vývoj modelu neurálního vykreslování s dobrou schopností generalizace stále pokračuje [46]. V poslední době jsou velice populární právě NeRF modely [47].

4 ANALÝZA PROBLÉMU

Jak bylo zmíněno v kapitole 2, použití řádkové kamery je v některých aplikacích výhodnější oproti kamerám s plošnými senzory. Problém nastává, pokud se snímaný objekt pohybuje i v horizontálním směru, tedy kolmo k směru snímání. Záznam řádku spojitě nenavazuje na předcházející nebo následující řádek a vzniká tak nežádoucí deformace. Objektem zájmu v této práci je živá ještěrka, která při snímání řádkovou kamerou dýchá a snímek deformuje. Zároveň může dojít vynechání řádků v důsledku řádkového zásobníku (obr. 2).

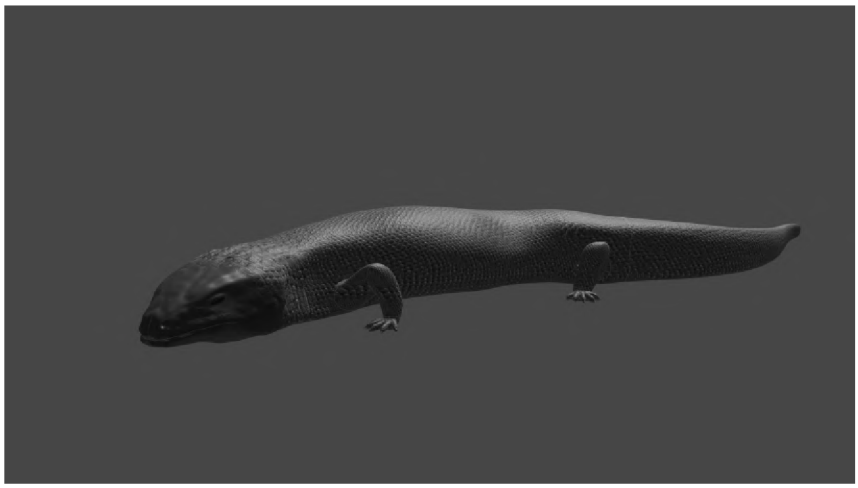
Obr. 12: Skutečný výstup řádkové kamery, ještěrka obecná

Nejbližší podobný problém se vyskytuje u aplikací zobrazovacích technik ve zdravotnictví [8, 9]. Při snímání orgánů je uvažován jejich minimální pohyb, větší množství pohybu způsobuje deformaci snímku. Kompenzace deformace vychází z informace 3D zobrazení, postupně transformované do nižších dimenzí v podobě 2D snímků a shluků bodů v grafu. Po kompenzaci dojde k jejich zpětné transformaci do 3D. Objekt a chyba snímání jsou podobné danému problému, ale metoda záznamu se liší.

Snímání sondou elektronového mikroskopu má podobný charakter záznamu jako řádková kamera [36]. Přítomné jsou deformace typu nelineární posuv, vynechání řádků, náhodný šum a rotace. Snímá se periodická struktura, která neodpovídá

našemu objektu zájmu, živočich v pohybu. Způsob kompenzace vychází ze snímků pořízených s fázovým posuvem.

Uvažovaný problém je tedy velice specifický. Podobné problémy existují, ale jejich řešení vychází z více snímků, které jsou fázově posunuté nebo informace ze záznamu ve vyšší dimenzi.



Obr. 13: Model ještěrky [29]

Výstup řádkové kamery (obr. 12) je deformovaný v důsledku dýchání ještěrky, při něm dojde k roztažení hrudního koše ve dvou osách. V ose x (obr. 20) a v menší míře i ose y (viz sekce 6.1). Z dostupných reálných snímků ještěrky byla potvrzena závislost aktuálního psychického rozpoložení a frekvence dýchání. Klidná ještěrka nevykazuje příliš velkou frekvenci hlubokého výdechu a nádechu ani mělkého dýchání. Rozrušená ještěrka kombinuje mělké a hluboké dechy, každopádně frekvence střídání těchto typů dýchání je těžko popsatelná. Další tělesné procesy způsobující pohyb torza ještěrky nebudeme uvažovat.

Osvětlení skutečné scény je realizováno jako liniové světlo pod úhlem. Zaručuje tak vhodnou intenzitu pro krátké expoziční časy a zároveň neprodukuje nechtěné odlesky v oblasti zájmu, pouze na končetinách ještěrky. Zadání specifikuje přítomnost a úpravu datové sady uživatelem. Zde uvažujeme jako datovou sadu ještěrky vyskytující se na území ČR. Hlavní rozdíly tvoří textury šupin a barvy vzorů šupin. Generátor by měl být schopen generovat nové textury a barvy specifické pro daný typ ještěrky.

Cílem je vytvořit generátor snímků, který simuluje výše popsanou deformaci, dle používaných metod z kapitoly 3. K dispozici nemáme dostatečně velký dataset reálných snímků reprezentující snímek pořízený řádkovou kamerou. Metody kompozice obrazu proto nelze použít, stejně jako kterákoliv metoda vycházející z reálných dat. Dostatečně detailní 3D model by měl posloužit jako náhrada skutečné ještěrky (obr. 13), a to s reálným nebo fiktivním pozadím.

Důležitou vlastností generátoru je variace generovaných dat. Scéna by se měla měnit, ale zároveň zůstat blízko k realitě (rozdíl domén). Osvětlení vnáší do scény jistou variaci, díky přítomnosti více jak jednoho zdroje světla a dalších faktorů (nechtěný pohyb objektů ve scéně, otřesy v okolí scény atd.). Intenzita a barva světla se mohou měnit.

Datová sada a její variace souvisí se simulovanou ještěrkou. Generátor by měl vytvářet syntetické textury o vysokém rozlišení s variací tvarů šupin, barevných vzorů a barev kůže specifického druhu. Deformace ještěrky je další část generátoru, ve které se značně projeví variace. Jedná se o další důvod proč byly zvoleny metody generace syntetických dat z 3D modelů. Vynechávání řádků v důsledku zásobníku kamery není dále uvažováno.

Dalším cílem práce a hlavní výhodou generátorů syntetických dat je automatické označování generovaných dat. To může být realizováno označením typu snímku (chybný referenční) do pixelů dat (text ve snímku) nebo v názvu souboru.

5 ŘEŠENÍ FUNKCÍ GENERÁTORU

Z analýzy problému (viz kapitola 4) víme o několika důležitých funkcích, které by měli být v generátoru implementovány (obr. 5). Valná většina se týká variace generovaných dat a označování snímků. Zde jsou uvedeny příklady možných způsobů a algoritmů jak realizovat tyto funkce.

5.1 Simulace dýchání

K realizaci animace mimo software k tomu určený by šlo využít exportu modelů reprezentující každý snímek. Program nahraje pro každé nové vykreslení jemu odpovídající model z Blenderu [15]. Toto řešení funguje a je jednoduché, jelikož veškerou interpolaci modelů animace realizuje Blender, ale řešení není příliš vhodné vzhledem k velikosti a detailu 3D modelu. Ukládání a čtení modelů je pomalejší než jeho výpočet pomocí interpolace a vykreslení.

K vykreslování jednotlivých snímků v sekvenci animace dýchání lze postupovat podobně jako v Blenderu. Pro případ videa je každý snímek vykreslen zvlášť. Mezi vykreslením snímků dochází k interpolaci modelu a změnám parametrů scény. Je třeba aspoň dvou modelů (výdech a nádech) mezi kterými interpolujeme.

Lineární interpolace je nejjednodušší způsob, jak realizovat simulaci dýchání mezi snímky. Je třeba znát číslo aktuálního snímku x_{curr}, předchozího x_{old} a následujícího x_{new}. Modely v rovnici reprezentuje y_0, y_1.

$$y_{curr} = y_0(1 - \frac{x_{curr} - x_{old}}{x_{new} - x_{old}}) + y_1(\frac{x_{curr} - x_{old}}{x_{new} - x_{old}}) \tag{1}$$

Další metoda interpolace je Bézierovou křivkou [1]. Mezi body y_0 a y_1 vložíme například další dva body, jedná se tedy o kubickou křivku. Aktuální model y_{curr} pro snímek x_{curr} získáme z (2). Hodnota t_i by neměla překročit hodnotu jedna [1].

$$y_{curr} = [1, t_i, {t_i}^2, {t_i}^3] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \cdot \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{2}$$

Zvýšení (snížení) hloubky nádechu jde realizovat jako rozdíl obou modelů, který násobíme pozitivním nebo negativním číslem a přičtením k modelu výdechu (3) a (4). Načtený model je v podstatě bod na přímce. Jednotlivé modely můžeme jednoduše manipulovat operacemi (sčítání, odčítání) s modely nebo hodnotami.

$$y_0 = y_0 + (y_1 - y_0)B_{výdech} \tag{3}$$

$$y_1 = y_0 + (y_1 - y_0)B_{nádech} \tag{4}$$

5.2 Generace textur s velkým rozlišením

K účelu generování nových textur existuje mnoho aplikací neuronových sítí [47] a sofistikovaných algoritmů generující tzv. bezešvé (seamless) textury, které na sebe přesně navazují. Jejich postupným skládáním dostaneme texturu větší velikosti.

Pokud takové textury nejsou dostupné, lze využít algoritmů, které zvětšují rozlišení textury (AI upscaling), nebo je naopak skládají (velká textura se skládá z menších), i když na sebe textura nenavazuje. Jedním z nejjednodušších algoritmů pro vytvoření větší textury z menší je algoritmus prošívání textur (1). Veškeré informace v této sekci vycházejí z [13, 37].

Algoritmus 1: Prošívání textur [37]

Vstup: Vstupní textura I_0, Velikost vzorku w_p Velikost překrytí w_o, Tolerance ε, Poměr výstup/vstup R

Výstup: Syntetizovaná textura I_s

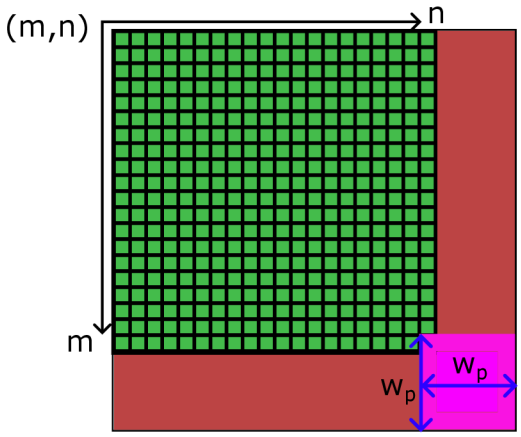
1 Inicializace I_s náhodným vzorkem
2 **Pro každý** *vzorek* $P_{old} \in I_s$ **proveď:**
3 Vyber vhodný vzorek $P_{in} \in I_0$ dle algoritmu (2)
4 Spočítej minimální cestu mezi překrytím P_{old} a P_{in} dle algoritmu (3)
5 Vytvoř vzorek P_{new} překrytím (prošitím) P_{old} a P_{in} v oblasti překrytí (5)
6 Zaměň P_{old} za P_{new} v I_s

Vstupem algoritmu je textura I_0. Velikost vzorku w_p určuje část vstupní textury I_0, ze které můžeme vybírat vzorky (obr. 14), uvažujeme čtvercový výstřižek, který označujeme jako P_{in}. Každý zelený čtverec, reprezentuje (m, n) počátek vzorku. Z červené oblasti nelze čerpat bez změny velikosti w_p.

Velikost překrytí w_o, určuje procento vzorku P_{old}, které zasahuje (překrývá) vzorky okolo něho (obr. 15). Parametr tolerance určuje množství vhodných vzorků P_{in}, ze kterých náhodně vybíráme. Poměr výstupu a vstupu R, určuje velikost syntetizované textury I_s, má největší dopad na čas výpočtu. Inicializace I_s (1) spočívá v náhodném výběru prvního vzorku P_{in} a jeho umístění do levého horního rohu I_s, kde $(m, n) = (0, 0)$.

Rovnice (5) z [37] popisuje tvorbu nového vzorku syntetizované textury I_s. Matice M má hodnoty mezi nulou a jedna, případně je binární. Definuje oblast ve které je aktuálnímu vzorku P_{new} přiřazena hodnota z P_{old}. Dle názvu P_{new} a P_{old} sdílejí stejné souřadnice v I_s. Inverze M má stejnou funkci, přiřazuje P_{in} do P_{new}. Vzorek P_{in} je vhodně vybraný výstřižek vstupní textury I_0.

$$P_{new} = MP_{old} + (1 - M)P_{in} \tag{5}$$



Obr. 14: Možný výběr počátku vzorku P_{in} (fialově) v I_0, (zeleně)

Algoritmus 2: Výběr vzorku [37]

Vstup: Vstupní textura I_0, Aktuální vzorek P_{old}, Velikost vzorku w_p, Tolerance ε, Binární matice Q

Výstup: Pozice (m, n), souřadnice vzorku P_{in} v I_0

1 Vypočítej matici vzdáleností na druhou D mezi P_{old} a všemi $P_{in} \in I_0$ dle (6) nebo (7)

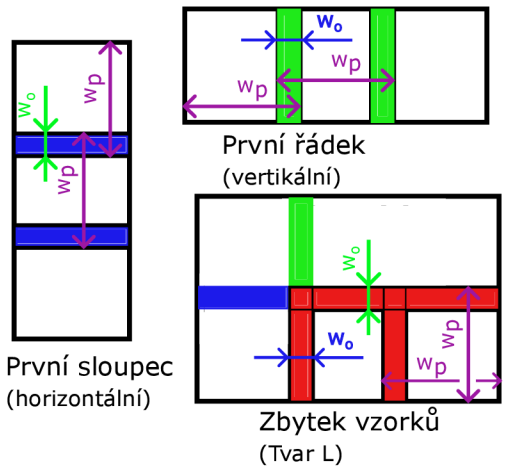
2 $d_{min} \leftarrow min\ D$

3 Vyber rovnoměrně pozici (m, n) z množiny kandidátů $\{(k, l), D(k, l) < (1 + \varepsilon)d_{min}\}$

Algoritmus (2) nahrazuje náhodný výběr vzorku z I_0 kritériem vzdálenosti. Vybraný vzorek P_{in} o velikosti w_p x w_p má levý horní roh umístěn v řádku m a sloupci n. Binární matice Q určuje překrytí o velikosti w_o mezi P_{in} a P_{old}, pokud je $(Q(i, j) = 1)$. Možné typy překrytí jsou uvedeny v (obr. 15). Kromě prvního řádku a sloupce mají všechny vzorky P_{old} oblast překrytí tvaru L. Jediný vzorek, který nezasahuje do dalších je v $(m, n) = (0, 0)$. Tento fakt zabraňuje paralelnímu výpočtu P_{old}, až na výjimky.

Vzorky P_{old} s počátkem $(m, n) \neq (0, 0)$ jsou definovány částečně z jiného vzorku. Pro vzorky v prvním řádku se jedná o levý sousední vzorek $P_{old}(m, n - 1)$, proto nelze provést jejich paralelní výpočet. Pro vzorek v prvním sloupci existuje závislost mezi dvěma vzorky nad ním, $P_{old}(m-1, n), P_{old}(m-1, n+1)$. Vzorek nové řady může být paralelně syntetizován jakmile jsou v řadě nad ním syntetizovány alespoň dva vzorky. Řádky mohou být zpracovány paralelně při splnění této podmínky, ale závislost vzorků mezi sebou vynucuje jejich sekvenční výpočet ve stejném řádku m [37].

$$D = ||P_{old}||_2^2 - 2\Gamma(P_{ext}, I_0) + \Gamma(Q_{ext}, {I_0}^2) \tag{6}$$



Obr. 15: Možné tvary překrytí vzorků v I_s [37]

$$D(m,n) = \sum_{i,j=0}^{w_p-1} Q(i,j)(P_{old} - I_0(m+i, n+j))^2 \tag{7}$$

Rovnice (6) udává výpočet matice vzdáleností na druhou. Nahrazuje (7) s využitím rychlé Fourierovi transformace [28]. Dolní index P_{ext}, Q_{ext} značí rozšíření P_{old}, Q na velikost I_0 doplněním nul. Operátor Γ značí korelaci. $\Gamma(Q_{ext}, {I_0}^2)$ lze vypočítat předem, existují pouze tři variace Q (obr. 15). Výpočet se tak zjednoduší na energii $||P_{old}||_2^2$, Eukleidovská norma, konkrétně suma všech pixelů umocněných na druhou v aktuálním P_{old} a korelaci $\Gamma(P_{ext}, {I_0}^2)$.

Algoritmus (3) hledá optimální řez (cestu) γ v oblasti překrytí. Zde existuje analogie s prošíváním, kdy jsou dva kusy (P_{old}, P_{in}) sešity dohromady pomocí matice M, která má hodnoty dané cestou γ. T_h a T_v slouží pro jednoduší zpětné hledání cesty bez dalších výpočtů [37].

5.3 Texturové mapy

Textura sama o sobě určuje barvu a vzhled modelu. Označuje se jako difuzní mapa (viz difuzní komponenta osvětlení) nebo albedo mapa [27]. Ostatní efekty (iluze hloubky, lesk a další) se často realizují dalšími mapami [2]. Mapy mohou být kombinovány do jedné (difuzní) procesem, který se označuje jako baking. Výhodou bakingu jsou menší nároky na detail modelu (méně polygonů), ale se stejným detailem textur jako u většího modelu.

Výšková mapa

Textura může obsahovat informaci o výšce v jednom z kanálů, například alfa kanál nebo jeden z RGB, který tuto informaci dobře popisuje. Pokud žádný kanál nevyhovuje lze využít algoritmů umělé inteligence, které informaci o výšce dokážou extraho-

Algoritmus 3: Minimální cesta mezi překrytím [37]

Vstup: Vzorek P_{old}, Vzorek P_{in}, Velikost vzorku w_p, Velikost překrytí w_o

Výstup: Cesta skrz překrytí γ

1 Vypočítej $e(i,j) = (P_{in}(i,j) - P_{old}(i,j))^2$

2 **Switch** *Typ překrytí* **:**

3 **Pokud** *vertikální* **:**

`/* Vertikální minimální kumulativní chyba` E_v `: */`

$E_v(w_p - 1, j) = e(w_p - 1, j), \quad j \in \{0, ..., w_o - 1\}$

4 **Pro každé** $i \in \{w_p - 2, ..., 0\}$ **proveď:**

5 $E_v(i,j) = e(i,j) + min(E_v(i+1, j-1), E_v(i+1, j), E_v(i+1, j+1)),$ $j \in \{0, ..., w_o - 1\}$

6 $T_v(i,j) = \text{argmin}(E_v(i+1, j-1), E_v(i+1, j), E_v(i+1, j-1)),$ $j \in \{0, ..., w_o - 1\}$

7 Urči $j^* = \text{argmin}_j(E_v(0, j))$

8 Najdi zpětně cestu s počátkem v $\gamma_{w_p - 1} = (0, j^*)$ $\gamma_i = T_v(\gamma_{i+1}),$ $i \in \{w_p - 2, ..., 0\}$

9 **Pokud** *horizontální* **:**

`/* Horizontální minimální kumulativní chyba` E_h`: */`

$E_h(i, w_p - 1) = e(i, w_p - 1), \quad i \in \{0, ..., w_o - 1\}$

10 **Pro každé** $j \in \{w_p - 2, ..., 0\}$ **proveď:**

11 $E_h(i,j) = e(i,j) + min(E_h(i-1, j+1), E_h(i, j+1), E_h(i+1, j+1)),$ $i \in \{0, ..., w_o - 1\}$

12 $T_h(i,j) = \text{argmin}(E_h(i-1, j+1), E_h(i, j+1), E_h(i+1, j+1)),$ $i \in \{0, ..., w_o - 1\}$

13 Urči $i^* = \text{argmin}_i(E_v(i, 0))$

14 Najdi zpětně cestu s počátkem v $\gamma_{w_p - 1} = (i^*, 0)$ $\gamma_j = T_h(\gamma_{j+1}),$ $j \in \{w_p - 2, ..., 0\}$

15 **Pokud** *tvar L* **:**

16 Vypočítej vertikální minimální kumulativní chybu E_v

17 Vypočítej horizontální minimální kumulativní chybu E_h

18 $i^* = \text{argmin}_i(E_v(i,i) + E_h(i,i) - e(i,i)), \quad i \in \{0, ..., w_o - 1\}$

19 Najdi zpětně vertikální část γ, počátek (i^*, i^*) $\gamma_i = T_v(\gamma_{i+1}),$ $i \in \{w_p - i^* - 2, ..., 0\}$

20 Najdi zpětně horizontální část γ, počátek (i^*, i^*) $\gamma_j = T_h(\gamma_{j+1}),$ $j \in \{w_p - i^*, ..., 2(w_p - i^*) - 1\}$

vat přímo z RGB hodnot textury. Existují i jednodušší metody aproximace výškové mapy, opět vycházejí z RGB hodnot textury. Vzorec $h = 1 - (1 - r)(1 - g)(1 - b)$, kde h označuje hodnotu výšky a r, g, b hodnoty RGB kanálů pixelu. Další možností je aproximovat výšku pomocí maxima z RGB kanálů, případně průměru těchto hodnot [2].

Normálová mapa

Normálová mapa vytváří iluzi hloubky (textura se jeví jako 3D), její výpočet vychází z výškové mapy. Pro získání hodnot (n_x, n_y, n_z) je třeba vypočítat normalizovaný vektor $(-\frac{\partial h}{\partial x}0.5+0.5, -\frac{\partial h}{\partial y}0.5+0.5, 1)$ [2]. K aproximaci výpočtu parciálních derivací lze použít Sobelův operátor pro složku x a y (obr. 16) nebo Scharr operátor (obr. 17).

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Obr. 16: Sobelův operátor, kernel 3×3

$$\begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix}$$

Obr. 17: Scharrův operátor, kernel 3×3

Mapa zvýraznění

Mapa je vytvořena aproximací z difuzní nebo výškové mapy úpravou kontrastu. Hodnota zvýraznění se využívá ve výpočtu leskové komponenty osvětlení. Hodnoty blízko maximální (např. 255) zvyšují efekt lesku, zatímco malé hodnoty ho utlumují. Změna kontrastu lze realizovat pomocí prahu, který přiřazuje hodnotám pod ním a nad minimální, maximální hodnoty. Práh může být globální nebo uvažován lokálně dle hodnot pixelů v lokálním okolí [2]..

5.4 Model osvětlení

Phong model je jedním z nejjednodušších modelů osvětlení [27]. Skládá se ze tří komponent, okolní, difuzní a leskové.

Okolní komponenta

Globální tmu ve scéně lze vytvořit vynásobením všech RGB komponent pixelů nulou. Implementace okolní komponenty je podobná, ale nulu nahradíme malou hodnotou

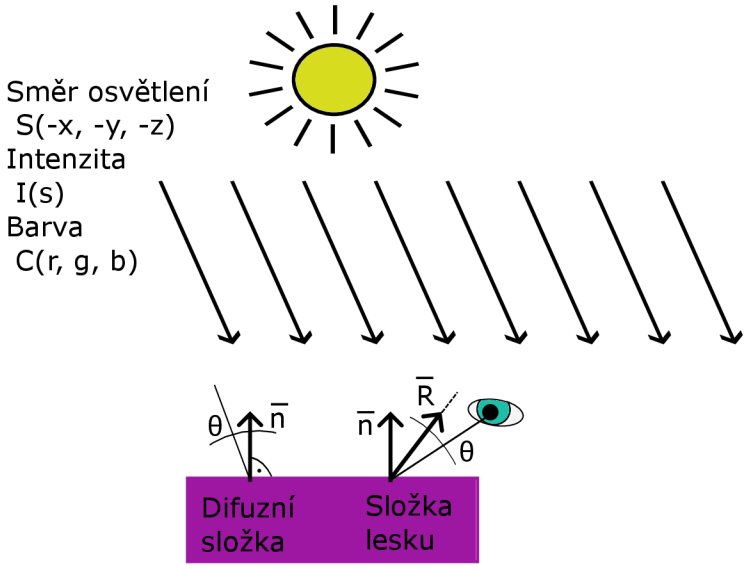
například 0.2 (obr. 24). Okolní komponenta simuluje světlo scény s jiného než hlavního zdroje osvětlení [27].

Difuzní komponenta

Simuluje dopad světla na materiál a jeho výslednou barvu, má tak největší vizuální efekt. Čím víc je stěna objektu nakloněna směru paprsků, tím vyšší hodnota, kterou jsou násobeny RGB komponenty. Cílem je spočítat úhel Θ v (obr. 18), který popisuje náklon světla a stěny objektu. Úhel získáme skalárním součinem normalizovaných vektorů, normály $\bar{n}$ a směru světla. Směr je parametr směrovaného osvětlení, není tak nutné dalších výpočtů k jeho získání [27].

Komponenta odlesků

Komponenta odlesků závisí na směru pohledu kamery (obr. 18) a na $\bar{R}$, normalizovaném vektoru odrazu světla. Opět úhel Θ určuje velikost hodnoty, kterou násobíme RGB komponenty textury. Rozdíl mezi pozicí objektu a kamery reprezentuje směr pohledu. Odraz $\bar{R}$ lze spočítat jako $\bar{R} = \bar{S} - 2(\bar{n} \cdot \bar{S})\bar{n}$. Úhel Θ získáme stejným způsobem jako u difuzní komponenty, ale s vektory odrazu a směru pohledu kamery. Příspěvek odlesků je tak roven násobku Θ, vlastností materiálu (lesk), síle světla a hodnotě mapy zvýraznění 5.3.



Obr. 18: Směrované osvětlení [27].

Reálná scéna využívá osvětlení liniovým světlem. Jelikož simulace nevykresluje snímky řádek po řádku, lze simulovat skutečné osvětlení scény pomocí směrovaného čtvercového světla (obr. 18). Paprsky jsou paralelní a každý objekt je osvětlován stejně. Parametry osvětlení jsou směr se souřadnicemi (x, y, z), barva s hodnotami (r, g, b) a intenzita osvětlení s [27].

5.5 Realizace barevných vzorů na kůži

Na skutečné ještěrce (obr. 12) se vyskytují barevné vzory specifické pro její druh. Tyto tvary by šlo generovat pomocí celulárního automatu se specifickými pravidly [14] nebo generativním modelem neuronové sítě [6]. Každopádně všechny možné přístupy musí být schopné popsat několik profilů (nastavení) reprezentující implementované druhy ještěrek a střídat mezi nimi. Další možnost jak vytvořit tyto vzory je nakreslením člověkem nebo generátorem, který napodobuje způsob nákresu štětcem do vrstev kresby vytvořené v nějakém softwaru k tvorbě digitálních kreseb.

Při pohledu na ještěrky (obr. 12) nebo další druhy vyskytující se na území ČR (Ještěrka obecná, zelená, živorodá, zední, travní) lze vidět typické barevné vzory. Vyskytující tvary (typy) lze rozdělit do kategorií: barva kůže daného druhu, pruhy, kaňky a body. Samozřejmě se jedná o zjednodušení, ale dále budeme uvažovat pouze tyto typy. Jejich hierarchie je dle uvedení. Na kůži specifické barvy aplikujeme pruhy, následně kaňky a jako poslední body. Tyto barevné vzory jsou reprezentovány pomocí barevných masek, které by měl generátor automaticky generovat. Aby jsme omezily oblast ve které se dané typy tvarů vykreslují nezbývá než omezit počet pixelů v masce. Taková maska musí být vytvořena nějakým pravidlem nebo ručně. Možná realizace tvorby barevných masek z masky vymezující jejich přítomnost je v algoritmu (4).

Algoritmus (4) popisuje generaci a aplikaci barevných masek. V prvních krocích vybereme z celkového počtu pixelů $|M_c|$ masky M_c pouze kandidáty. Jejich počet $|P|$ je dále omezen, kvůli velikosti použitých textur na n_g. Následně pro všechny kandidáty, pixely p, se souřadnicemi (m, n) v prázdné masce I_e provedeme vybarvení okolí (kruh, kosočtverec, šestihran atd.) a operaci eroze, dilatace dle hodnoty i_{val}. Barva je v rozsahu $c \in \{0, ..., 255\}$, ale pro správný výsledek je třeba vstup kde $c_{norm} = \frac{c}{100}$. Výslednou masku I_e vynásobíme s I_g. Výstupem je barevná textura $I_c(r, g, b, a)$.

Algoritmus 4: Kombinace barevné masky a textury

Vstup: Černobílá textura I_g, Barevná maska M_c o rozměrech M_x x N_y, Prázdná maska I_e, Barva druhu ještěrky $c = (r, g, b)$

Výstup: Textura $I_c(r, g, b, a)$

1 $n_P = |P|$, $P = \{p_0, ..., p_i, ..., p_n\}$, $\{p_i \mid p_i \in M_c \wedge p_i(r, g, b, a), a > 0\}$, $i \in \{0, 1, ..., |M_c| - 1\}, |M_c| = M_x \cdot N_y$

2 Náhodně vyber n_g, $n_g \in \{1, ..., n_P\}$

3 $i_{val} = 0$

4 **Pro každé** $i \in \{0, ..., n_g - 1\}$ **proveď:**

5 Náhodně vyber $p \in M_c$ a hodnotu w nebo $p_i \in M_c$, $w = sin(i_{val} \cdot 2\pi)$

6 $(m, n) = arg(p_i)$

7 V okolí o náhodné velikost r_o se středem v $I_e(m, n)$ přiřaď hodnotu c všem pixelům v tomto okolí.

8 **Pokud** $w >= 0$ **potom:**

9 Proveď operaci dilatace na okolí $I_e(m, n)$

10 **Jinak pokud** $w < 0$ **potom:**

11 Proveď operaci eroze na okolí $I_e(m, n)$

12 **Pokud** w *získáváme periodickou funkcí* **potom:**

13 $i_{val} += \frac{r_o}{n_g}$

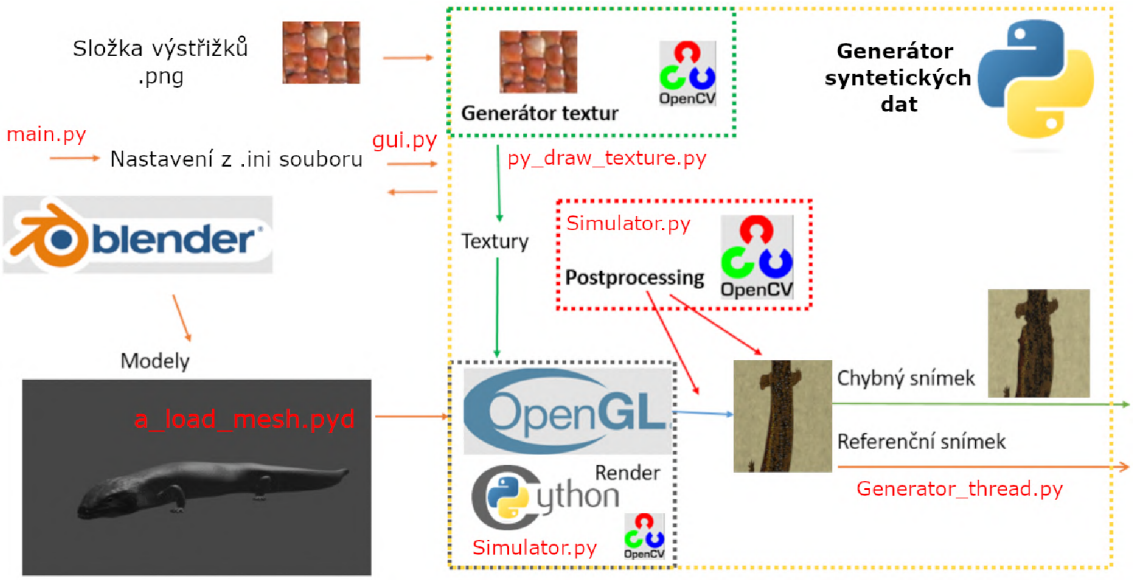
14 $I_c = I_g I_e$

6 GENERÁTOR SYNTETICKÝCH DAT

Schéma celého programu (obr. 19) má jako hlavní část generátor syntetických dat. Vstupem generátoru jsou modely ve formátu Wavefront, nastavení z .ini souboru a .png výstřižky textur šupin. Před spuštěním generátoru dojde k načtení .ini souboru, který určuje další chod programu. Sekce tohoto nastavení se dají shrnout na změnu rozlišení, animace, datové sady, počtu exekucí a postprocessing.

V prvních krocích generátoru dojde k syntetizování textur a texturových map tvořící ještěří kůži. Data textury a modelu se přesunou do programu OpenGL (vertex a fragment shader). Manipulace modelu, jako je animace a deformace pro variaci generovaných dat, jsou realizovány ve vertex shaderu. Fragment shader realizuje osvětlení, barvy a mapování textur na modely. Python program poskytuje pouze náhodné generování parametrů generátoru (viz sekce 6.3), které jsou opět posílány do OpenGL programů.

Funkce obou programů OpenGL realizují Render část v (obr. 19). Při každém vykreslení dojde k zachycení syntetického snímku, který je dále manipulován postprocessingem. Hlavní smyčka běží dle nastaveného počtu exekucí, s každým novým cyklem dojde k náhodnému nastavení parametrů generátoru. Poslední krok programu přiděluje označení snímků jako referenční nebo chybný. Snímky se ukládají dle nastavené cesty v .ini souboru nebo změnou v GUI.



Obr. 19: Schéma implementovaného generátoru dat.

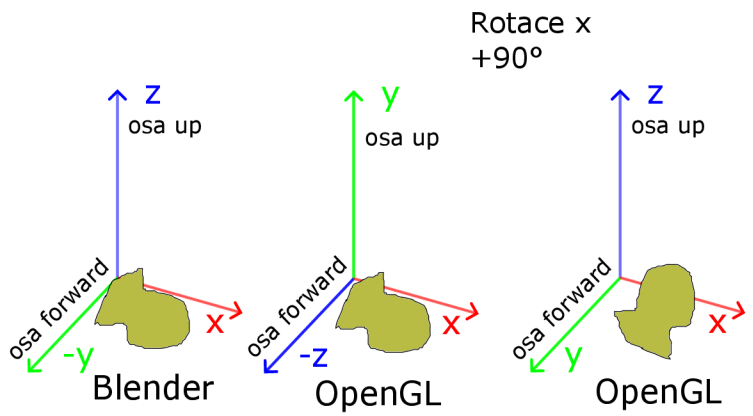
Software je realizovaný pomocí jazyka Python 3.9, který byl vybrán na bázi Blender API kompatibility a zkušenosti autora práce. Volby knihoven se odvíjí od použitého jazyka a kompatibility použitých verzích. Volba Blenderu k modelování (viz sekce 6.1) je opět dle zkušenosti autora a toho, že je software volně dostupný.

Některé funkce programu jsou implementovány současně i v Cythonu, konkrétně generace textur a barevných masek. Výchozí nastavení generátoru jich využívá. Zda použít Python nebo Cython implementaci lze nastavit v .ini souboru. Formát .ini byl opět zvolen dle zkušenosti autora. Funkce implementované v Cythonu zkracují exekuční čas, obzvláště při nastavení více položek datové sady. Pro maximální datovou sadu bylo pomocí testování zjištěno zkrácení exekučního času okolo 10 sekund. Efekt u menší datové sady klesá a exekuce s jedním kandidátem v sadě je časově stejná. Generace textur a barevných masek je tak urychlena, ale funkce jsou sestaveny pouze pro Windows OS.

Software generátoru existuje ve dvou verzích, Python implementace a verze zkompilovaná pomocí kompilátoru nuitka. Z důvodu kompatibility s kompilátorem byla použita verze OpenCV 4.5.3.56. Ze stejného důvodu je GUI realizováno v PySide6. Zároveň bylo nutné zkompilovat a přidat celou knihovnu OpenGL do archivu, což značně ovlivňuje velikost programu. Archiv je dostupný na GitHubu uvedený v příloze B.

6.1 Modely ještěrek

Původní volně dostupný model z [29] byl vytvořen v softwaru 3ds Max. Tento export byl chybně načítán do OpenGl. Zdroj tohoto problému nebyl zjištěn, ale nejspíš se jednalo o prohození souřadnicového systému (obr. 20). Po nahrání původního modelu do Blenderu (obr. 4) a opětovného exportu do formátu Wavefront (.obj) byl problém vyřešen. Je nutné uvézt rozdíl používaného souřadnicového systému mezi oběma softwary (obr. 20). Nastavení exportu z Blenderu umožňuje nastavit osu up a forward tak, aby rotace modelu v OpenGL pro pohled shora nebyla nutná.



Obr. 20: Souřadnicový systém Blenderu, OpenGL a pohledu shora [27]

Jako vstup generátoru slouží dva modely, reprezentují výdech a nádech ještěrky. K aproximace těchto modelů slouží video z rentgenových snímků dýchání va-

Výpis 6.1: Příklad dat ve formátu Wavefront (.obj)

```
www.blender.org
mtllib lizard_inhale.mtl
o lizard_inhale_reference
v -0.000946 0.000616 0.002194
vn -0.8516 0.0201 0.5237
vt 0.200600 0.324000
f 77/79/77 10/10/15 10/14/13 74/79/77
```

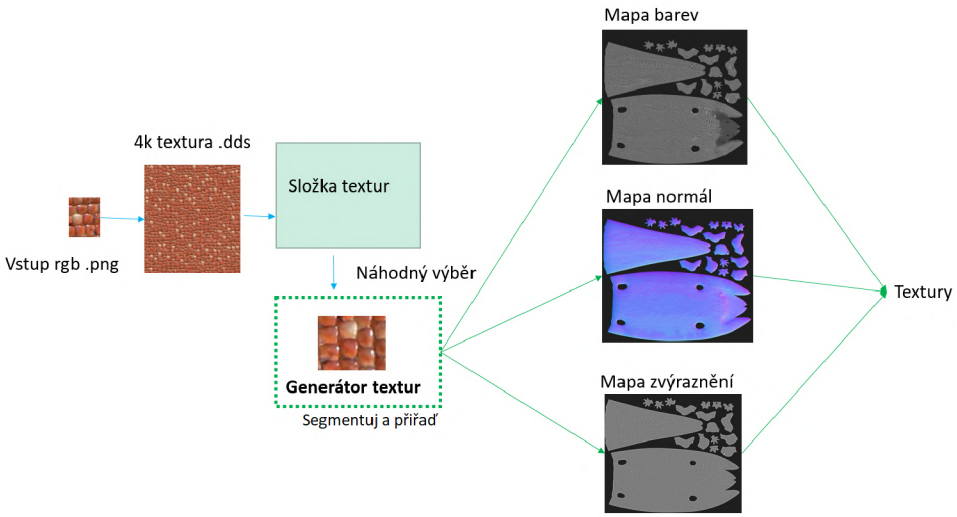
rana [51]. Oba tyto modely byly vytvořeny v Blenderu a vyexportovány ve formátu Wavefront. Generátor také načítá jednoduchý model podložky s texturou gaussovského šumu vytvořený v Blenderu. Podložka realizuje pozadí snímků.

Z Wavefront souboru potřebujeme následující informace: vrcholy objektu, texturové souřadnice a normály. Pozice vrcholů, řádky začínají písmenem v, obsahují x, y, z souřadnice. Vrcholy jsou v pořadí od zadní části modelu v protisměru hodinových ručiček. Souřadnice textur jsou oddělené od x, y, z souřadnic, jsou označené jako vt s horizontální a vertikální hodnotou (u, v). Oddělení dovoluje transformaci geometrie bez změny mapování textury na model. Normály modelu začínají písmeny vn s nx, ny, nz hodnotami. Model je běžně reprezentován seznamem stěn. Každá z nich je definována na řádcích začínající písmenem $f_{v,vt,vn}$ a obsahuje hodnoty v, vt, vn. Jednotlivé čísla značí indexy řádku v souboru (seznamu těchto hodnot). Při čtení jednoduše celý model nahrajeme do jednorozměrného pole ve formátu $x, y, z, u, -v, n_x, n_y, n_z$. OpenGL má opačný směr vertikální osy souřadnic textur, proto negace druhé souřadnice -v [51].

6.2 Generování textur a masek

Originální textura modelu není příliš detailní. Jako zdroj nových textur byl použit snímek kůže ještěrky obecné z [41], a jeho rozlišení zvětšeno pomocí online generátoru [50]. Následně byl snímek importován do Blenderu jako štětec, pro vytvoření detailní textury šedi (obr. 13). Tato textura je vždy použitá pro první výstup generátoru.

K této textuře jsou doplněny další texturové mapy, konkrétně mapa zrcadlových odrazů (specular map) a normálová mapa (normal map) vytvořené pomocí softwaru [7]. Obě slouží k realizaci efektů osvětlení. Další variace textur mají tento proces automatizován.



Obr. 21: Obecná posloupnost operací generátoru textur

Automatické generování textur šupin

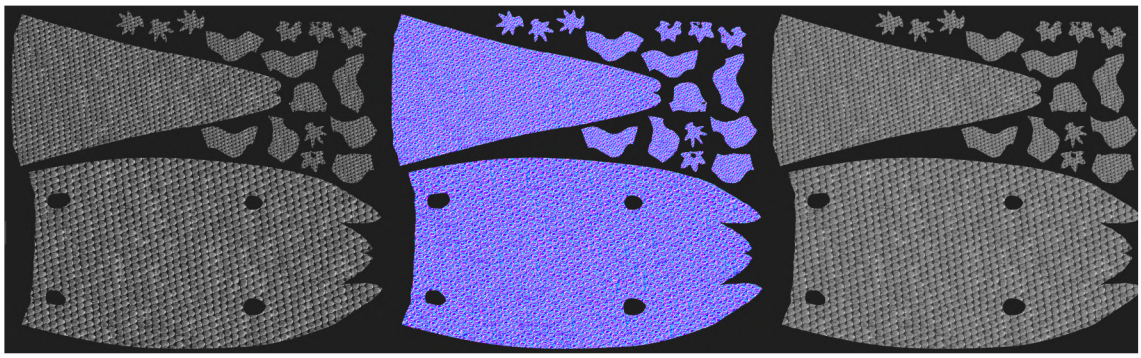
Vstup prvního kroku dle (obr. 21) je textura šupin o menší velikosti než výstupní. Výstup je syntetizován pomocí algoritmu prošívání textur (1), konkrétně jeho paralelní varianta [37]. Vstupní textury (výstřižky) jsou z fotek druhů ještěrek, které byly získány z [41]. Algoritmus vytváření difuzních map je realizován v souboru quilting process ve složce texture generator. Výstupem jsou textury s velkým rozlišením (záleží na nastavení v souboru) ve formátu .dds, kvůli ušetření místa na disku. K tomuto účelu vyžaduje program knihovnu Wand, která realizuje vazbu na software ImageMagick. Osmdesát 4K textur zabírá okolo 500 MB. Textura .dds formátu má velikost okolo 6MB, zatímco původní formát .png má okolo 35 MB.

Generování texturových map

Kromě textur šupin (albedo, difuzní mapa) je generátor schopen generovat a pracovat s mapou normál a zvýraznění (obr. 22). Obě tyto mapy slouží k simulaci efektů osvětlení. V prvním kroku dojde k segmentaci původní textury na jednotlivé části nohou, ocasu (obr. 22). Syntetizovaná textura je náhodně vybrána ze složky textur (obr. 21). Každé segmentované části je přiřazena část syntetizované textury. Vzniklá difuzní (albedo) mapa slouží jako zdroj informace pro tvorbu mapy normál a zvýraznění.

Generátor využívá výpočet výšky pomocí maxima RGB kanálů (viz kapitola 5.3). K realizaci výpočtu derivace využíváme knihovnu OpenCV, která pro kernel 3×3 používá operátor Scharr, varianta x a y na (obr. 17), který v některých situacích funguje lépe než Sobelův operátor [35]. Normalizace vektoru je mezi hodnotami (0, 1), ale výsledný obraz musí být násoben hodnotou 255, případně změnit normalizaci, aby byla textura viditelná.

Mapa zvýraznění je vytvořena aproximací z difuzní mapy. Generátor má nastavený práh okolo 95% z celkového maxima 255 (8 bitové barvy). Pixelům s hodnotou v tomto intervalu je přiřazena hodnota 255. Zároveň je všem pixelům mimo tento interval přidáno 10% maxima. Proces generování texturových masek je realizován v souboru py_draw_texture.py. Proces běží v pozadí na samostatném vlákně a předává textury hlavnímu procesu.



Obr. 22: Příklad nově generované masky albedo, normálová a zvýraznění (zleva)

Generace a aplikace barevných masek

Difuzní mapa je v odstínech šedi (obr. 22), ale s více kanály (RGBA). Alfa kanál je roven původní textuře, zbylé kanály mají stejné hodnoty, které jsou rovny odstínu šedi. Pro aktuální druh ještěrky jsou generovány barevné masky dle postupu v algoritmu (4). Původní zdroje masek, vstup (viz 4), byly vytvořeny v programu GIMP (ručně). Jako reference sloužily snímky z [41] pro všechny druhy ještěrek. Každý druh ještěrky je složen ze tří masek reprezentující pruhy, kaňky a tečky na difuzní mapě. V algoritmu (4) náhodný výběr nahrazuje $sin(i \cdot 2\pi)$. Periodická funkce aproximuje celistvost barvených míst, nebo mezery, které můžeme vidět na šupinách ještěrek (příloha A). Pro všechny masky je proces jejich generace realizován naráz, paralelně.

Barevné masky jsou aplikovány na difuzní mapu v OpenGL programu (fragment shader). V rámci zvýšení variace dojde k malé náhodné změně RGB kanálu barevné masky, následně dochází k násobení RGB hodnot masky a mapy, výsledek na (obr. 23). Pro případy kdy je výsledná barva příliš tmavá nebo světlá, dojde k ztracení odlesků nebo tmavých míst. V shaderu existuje globální práh světlé a tmavé barvy, který by měl tento problém odstranit.

Výchozí nastavení generuje barevné masky v draw_alpha.pyd souboru. Pokud je nastavení použití Cython funkcí vypnuto existuje Python implementace přímo v hlavním souboru Simulator.py.

Obr. 23: Výsledek násobení barevné masky a textury šupin po vykreslení

Osvětlení

Způsob jakým je scéna osvětlena při použití řádkové kamery z kapitoly 2 a analýzy problému 4, aproximuje směrované čtvercové světlo pod úhlem (obr. 18). V generátoru je zpracován Phong model 5.4, jelikož je osvětlení v reálné scéně relativně jednoduché. Okolní komponenta aproximuje světlo v okolí skutečné scény. Difuzní realizuje barvu kůže a lesková komponenta se dle nastavení směru osvětlení projevuje hlavně na končetinách modelu (obr. 24).

Normály získáme při čtení objektu (viz sekce 6.1), nebo pomocí normálové mapy 5.3. Odrazovou komponentu získáme v knihovně OpenGL funkcí $reflect()$. Implementace osvětlení je realizována v programu OpenGL (fragment shader) dle sekce 5.4 pro model ještěrky a podložky.

Obr. 24: Okolní, difuzní a lesková komponenta Phong modelu.

6.3 Parametry generátoru

6.3.1 Náhodné parametry

Model, textury a osvětlení jsou vykresleny na (obr. 23). Implementované parametry přispívající k zvýšení variace výstupu generátoru jsou: změna geometrie, barev a osvětlení. Hodnoty parametrů jsou náhodné bez uživatelského vstupu. Jejich generace je realizována v Simulator.py, zatímco aplikace variací probíhá v programech OpenGL (vertex shader, fragment shader).

Stručné shrnutí variace barev a osvětlení spočívá v náhodné změně jedné nebo více hodnot. RGB komponenty textury a osvětlení měníme odečtením nebo přičtením malé hodnoty k původní hodnotě, která je uchována v paměti během chodu programu.

Variace geometrie

V generátoru je implementována translace, zrcadlení, změna měřítka a ohyb modelu. Transformace měřítka a translace jsou realizovány násobením matice projekce (obr. 25), vytvořené pomocí knihovny Pyrr a vektoru souřadnic bodu $(x, y, z, 1)^T$. Všechny uvedené transformace se týkají pouze (x, y) souřadnic (viz kapitola 6.1 a 4). Důležité je zmínit, že OpenGL používá $(sloupec, řádek)$ notaci pro matice. Aby bylo uspořádání stejné s (obr. 25), je matice projekce transponována před předáním OpenGL programu.

Zrcadlení je implementováno negací původních souřadnic bodu $(-x, -y, z, 1)^T$. Generátor využívá možnost vykreslování pouze přední částí modelu, tedy pouze toho co vidí kamera (obr. 23), proto je nutné při zrcadlení přepínat mezi vykreslováním přední a zadní části modelu (glCullFace(GL_BACK), glCullFace(GL_FRONT)).

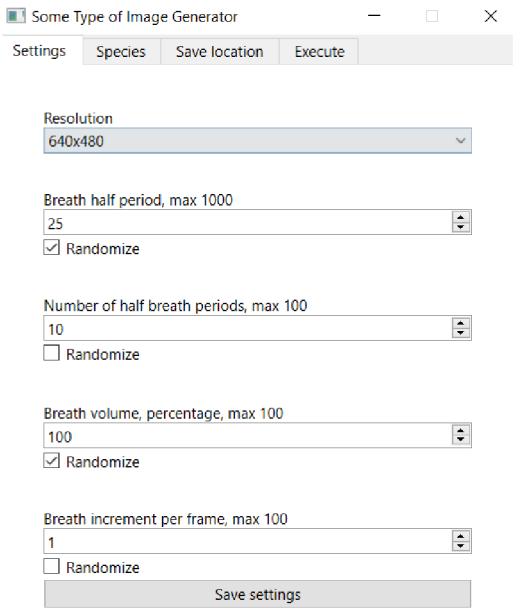
Ohyb realizuje periodická funkce jdoucí od hlavy až po ocas modelu ještěrky. První krok je získání souřadnic maxima a minima, tedy začátek hlavy a konec ocasu. Tyto hodnoty normalizují souřadnice bodů modelu, které slouží k výpočtu odchylky $d_x = Asin(2\pi f \cdot y_{norm} + \phi) + o$. Parametry funkce (amplituda A, frekvence f, fáze ϕ, offset o) jsou generovány náhodně v určitém rozsahu zjištěném experimentálně: $A\langle 0.0025\ ,\ 0.05)$, $f\langle 0.0001{,}1.5)$, $\phi\langle$-1,1), $o\langle 0,\ 0.005)$. Výsledná odchylka je přičtena k T_x.

$$\begin{bmatrix} S_x & 0 & 0 & T_x \\ 0 & S_y & 0 & T_y \\ 0 & 0 & S_z & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Obr. 25: Matice projekce, změna měřítka a translace

6.3.2 Uživatelský vstup

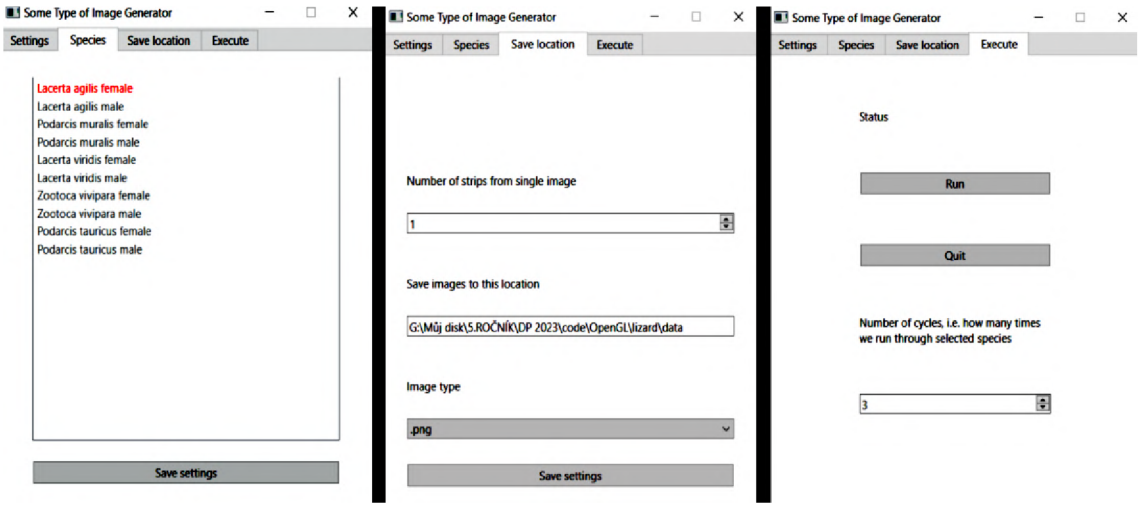
Generátor využívá knihovny PySide6 k vytvoření uživatelského rozhraní, které umožňuje nastavit parametry simulovaného dýchání, druhu ještěrek v datové sadě a nastavení uložení snímku. Nastavené hodnoty jsou uloženy a načítány z settings.ini souboru.



Obr. 26: Nastavení rozlišení a parametrů dýchání.

Parametry na (obr. 26) souvisí s nastavením simulovaného dýchání. Inkrement slouží k zvýšení míry interpolace mezi modely ještěrky, zrychluje tak dýchání nezávisle na nastavení půl periody. Náhodná generace přiřadí této hodnotě náhodnou hodnotu. Počet snímků v jedné půlperiodě určuje délku jednoho nádechu (výdechu). Náhodná generace způsobí náhodné přiřazení hodnot parametrům animace. Hloubka nádechu v procentech umožňuje zmenšit velikost modelu ještěrky reprezentující nádech, při náhodné generaci je této hodnotě přiřazena náhodná hodnota a uživatelský vstup slouží jako horní limit náhodné generace.

Nastavení datové sady vidíme na (obr. 27). Červeně vyznačené druhy budou použity při chodu generátoru. Pořadí použití druhu při generaci je stejné jako druhů uvedené v tomto seznamu. Nastavení uložení (obr. 27), dovoluje vygenerovaný snímek rozdělit na několik menších, pruhů. Dále lze nastavit kde soubor uložit a v jakém formátu. Dostupné formáty jsou stejné jako ty, které podporuje OpenCV (.png, .jpg, .bmp, .exr). Poslední nastavení, počet cyklů (obr. 27), určuje kolikrát za sebou bude generátor spuštěn, při novém cyklu dojde k náhodné generaci všech parametrů.



Obr. 27: Nastavení datové sady, uložení a spuštěni.

6.4 Animace a záznam dýchání

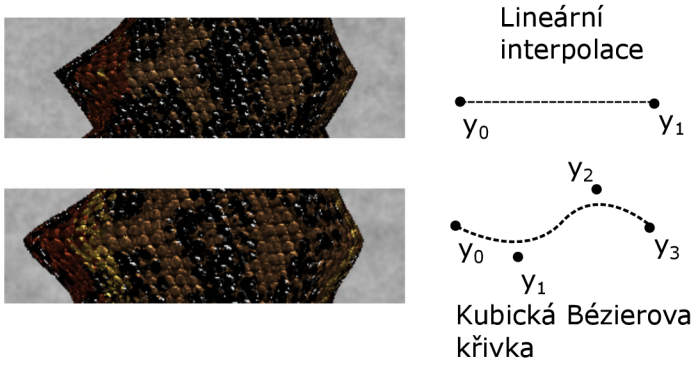
Namísto exportu nového modelu pro každý snímek nebo vzorkování z videa, využívá generátor interpolace mezi modely. Nový bod vzniklý interpolací reprezentuje model, který bude v daném snímku vykreslen y_{curr}. Dle lineární interpolace (1) je třeba znát číslo aktuálního snímku x_{curr}, předchozího x_{old} a následujícího x_{new}. V generátoru platí, že y_0 je model ve výdechu a y_1 nádechu. Toto platí pouze během první půlperiody. Dochází k přechodu z y_0 do y_1 (nádech), následně se modely prohodí a dojde k výdechu. Během chodu generátoru dochází k periodickému střídání modelů.

K simulaci mělkého a hlubokého typu dýchání (viz kapitola 4) lze použít uživatelské nastavení. Pokud je nastavena náhodná změna hloubky nádechu dojde i k náhodné změně mezi hlubokým a mělkým typem výdechu a nádechu (tj. během mělkého dýchání dojde občas i k hlubokému).

Výsledek lineární interpolace (1) lze vidět na (obr. 28). Jako náhrada je v generátoru implementována interpolace pomocí Bézierovy křivky (2). Generátor používá $t_i = \frac{x_{curr}-x_{old}}{x_{new}-x_{old}}$. Výpočet interpolace je opět realizován v programu OpenGL (vertex shader).

Záznam snímku

K simulaci záznamu řádkovou kamerou opět poslouží knihovna OpenGL. Generátor je schopen pracovat s $glfw$ oknem z nástrojů pro OpenGl, nebo snímkovým bufferem fbo. Okno $glfw$ tvoří kontext pro vykreslení a vidíme ho při chodu generátoru. Problém nastává při generaci snímků větších než je rozlišení monitoru ($glfw$ okna). Část snímku není vykreslena, uložené snímky mají okraje tvořeny černými pixely. Snímkový buffer tento problém řeší, ale nelze jej vidět, tedy nevidíme průběh ani-



Obr. 28: Interpolace modelu v generátoru.

Obr. 29: Generovaná deformace, postprocessing a skutečný snímek.

mace. Dvojí vykreslování v menším rozlišení do okna $glfw$ a skutečný záznam pro zachycení v fbo s větším rozlišením není implementován, je třeba volit mezi nimi v nastavení.

Při prvním vykreslení (první snímek cyklu aktuálního druhu ještěrky) zaznamenáme všechny pixely fbo nebo $glfw$ okna a uložíme referenční snímek (obr. 23). Další záznam má charakter scanneru (řádkové kamery), kdy z každého nového snímku zaznamenáme pouze jeden řádek o výšce jednoho pixelu. Jednotlivé řádky postupně vyplňují prázdný snímek dokud neskončí animace nebo počet řádků přesáhne rozlišení výšky snímku. V obou případech vykreslování končí a výsledný snímek se uloží jako chybný (obr. 29 vpravo).

Postprocessing

Jak bylo uvedeno v sekci postproressing (kapitola 3), k snížení rozdílu domén jsou snímku přidávány nechtěné vlastnosti. Ty simulují jejich přítomnost v důsledku

podmínek při jeho záznamu. Před uložením snímku generátor provede operace přidávající šum. Konkrétně ovlivnění náhodným šumem pro jednotlivé kanály, jejich součet simuluje barevný šum, a následně aplikace Gaussovského operátoru k odstranění ostrých hran (obr. 6). Obě operace byly realizovány pomocí knihovny OpenCV, vycházející z normálního rozdělení pravděpodobnosti.

Poslední operací je aplikace rozostření kontury ještěrky ve snímku. Pomocí prahování funkcí OpenCV vytvoříme binární obraz ještěrky odděleného od pozadí. Následně určíme konturu a její tloušťku. V oblasti definované konturou nahradíme pixely těmi z výrazně rozostřeného snímku Gaussovským operátorem. Výsledek všech operací vidíme vpravo na (obr. 29).

Označení snímku

Dle cílů práce měly být snímky označkované. Generátor při ukládání snímků vytvoří jméno, které reprezentuje některé vybrané informace. Z názvu lze určit, zda se jedná o referenční nebo chybný snímek. Zbytek jména slouží k uložení uživatelských parametrů (obr. 30).

Pořadí vygenerovaného snímku
n-tý pruh tvořící vygenorovaný snímek
Sníměk referenční (b), deformovaný (d)
HP - půlperioda (half period)
NP - počet period (number of periods)
BV - hloubka nádechu (breath volume)
PI - inkrement interpolace (per frame increment)
P - celkový počet pruhů rozdělení n, (partions)
NC - počet cyklů generátoru (number of cycles)
Formát snímku

1_2_b_HP50_NP6_BV0.25_PI1_P1_NC5_.png

Obr. 30: Způsob označení snímku.

7 DISKUZE VÝSLEDKŮ

Vytvořený software je prvotní realizací testující proveditelnost generátoru syntetických dat, pro účel vývoje algoritmů kompenzující zkreslení pohybem při snímání řádkovou kamerou. Z tohoto důvodu není implementace softwaru zcela optimalizovaná (paměťové nároky). Jelikož se jedná o první iteraci softwaru, data z něho nebyla doposud využita k žádnému vývoji. Jde tedy těžko zhodnotit použitelnost generovaných dat. Ve skutečnosti se může prokázat zcela mylná představa o závislosti některých vlastností ke kompenzaci deformace.

Hlavní nedostatek, který brání snadnému používání, je příliš složité nastavování parametrů generátoru (viz sekce 6.3). K tomuto účelu by bylo vhodné nahradit část GUI (viz sekce 6.3.2), interaktivní Bézierovou křivkou, která by umožnila nastavovat všechny parametry dýchání v jednom celistvém nastavení.

V příloze C nebo (obr. 29), jsou uvedeny reálné a syntetické data, které jsou výstupem vytvořeného generátoru dat. Při subjektivním porovnání zrakem autora vychází najevo rozdíl domén reálných a syntetických snímků. První nedostatek je v barevných maskách textur, které zcela neodpovídají barevným tvarům skutečných ještěrek. Řešením může být překreslení masek tak, aby byly více detailní (tečky a pruhy na těle ještěrky). Druhý rozdíl je v profilu deformace snímku, který má u syntetických dat větší variaci, než se objevuje u skutečných snímků (viz příloha C). Příčinou je nevhodné nastavení parametrů dýchání (půlperioda, hloubka dýchání viz 6.3.2), před spuštěním softwaru. Vhodné hodnoty musí být zjištěny pro každé rozlišení snímků zvlášť, což je velice zdlouhavé.

Střídání mělkého a hlubokého dýchání je další faktor ovlivňující profil deformace. Přechod mezi nimi je v generátoru náhodný, proto jsou některé snímky více podobné skutečným než jiné. Řešením je implementovat závislost mělkého a hlubokého dýchání, která se velice často střídá v oblasti blízko hlavy ještěrky a začátku hrudního koše. U nižších částí koše hloubka dýchání a deformace klesají. Tuto závislost lze simulovat pomocí kódu nebo změnou modelu reprezentující nádech. Nižší část torza se jednoduše zúží, což se projeví v interpolaci mezi modely menší deformací v nižší části modelu. Nevýhodou tohoto přístupu by bylo snížení variace profilu deformace.

Třetí rozdíl tvoří hladkost (rozmazání) profilu deformace. Skutečný snímek má hladší deformaci než ty syntetické. Řešením může být zvýšení bodů interpolace Bézierovi křivky, ale za cenu většího množství nastavovaných parametrů. Jednodušším řešením je zvětšení kernelu gaussovského operátoru v .ini souboru, což může být problém u generace dat s velkým rozlišením. U skutečných snímků není zcela jasné jak vypadá profil deformace bez šumu s vysokým rozlišením.

Syntetickým snímkům také chybí stíny. U skutečných snímků by neměli být přítomny, vzhledem k použitému typu osvětlení, ale okolní světlo vždy do jisté míry ovlivňuje scénu. Zároveň natočení ještěrky před kamerou nemusí být vždy stejné (viz sekce 6.1, osa y při pohledu shora), což může jednu stranu ještěrky udělat tmavší než druhou.

Generátor má mnoho částí, které lze v budoucích verzích vylepšit nebo přidat. Každý parametr generátoru ovlivňuje generovaný snímek s velkým množstvím nastavitelných kombinací. Hledání vhodných parametrů manuálním testováním je zdlouhavé. V generátoru snímků [40] používají automatické nastavení parametrů postprocessingu, což by ulehčilo nastavování zbylých hodnot.

Generování textur má problém v podobě nepříliš detailních výstřižků, které jsou součástí zvětšené textury. Textura je tak rozmazaná nebo negativně ovlivňuje vytváření texturových masek, které vychází z informace této textury (tmavší textury viz příloha A, C). Řešením by bylo vyřadit výstřižky (vzorky, viz sekce 5.2), s nízkým rozlišením, ale v takovém případě přijde generátor o diverzitu variace. Generování nekonečného množství nových textur lze realizovat například jako GAN model (viz sekce 3.1.1). Zde by opět muselo dojít k výběru výstřižků s vysokým rozlišením, které nemusí být dostupné v dostatečném množství.

Model ještěrky vyhovuje při interpolaci (sekce 6.4), ale jeho načítání zpomaluje chod programu. Při spuštění generátoru nejvíc času zabírá právě načítání modelů. Bylo by vhodné otestovat použití modelu s menším detailem (počtem polygonů), což by ovlivnilo i souřadnice textur, případně zvýšit počet modelů, které generátor může načítat.

V rámci přidání nových částí softwaru by bylo vhodné zmíněné problémy vyřešit, nebo využít Blender API jako náhradu OpenGL. Vykreslování v Blenderu zabírá více času než u vytvořené implementace, ale výsledný snímek je více realistický díky složitějším modelům osvětlení a vykreslování [33].

8 ZÁVĚR

Z poznatků teoretické části a analýzy problému byl vytvořen software simulující výstup řádkové kamery. Kamera simulace je umístěna v pozici nad ještěrkou, snímá pouze pohyb hrudního koše. Generátor produkuje data ve formě chybných a referenčních snímků, které jsou ukládány s označením typu snímku a vybranými parametry použitých při jeho generování (viz sekce 6.4). Toto označení je ve formě názvu souboru. Celý software je realizován pomocí rozhraní OpenGl, konkrétně zpracování pro Python 3.9 a novější.

Datovou sadu softwaru tvoří druhy ještěrek vyskytující se na území České republiky (ještěrka obecná, zelená, živorodá, zední, travní), jejich použití lze změnit v GUI softwaru. Nejvýraznější generované variace jsou transformace geometrie 3D modelu (ohyb, translace, zrcadlení) a generování nových textur. Generátor také náhodně mění barvu a intenzitu osvětlení.

Parametry nastavitelné uživatelem jsou: rozlišení snímku, nastavení parametrů dýchání, lokace uložení výstupních dat, datová sada ještěrek, počet cyklů generátoru, parametry animace, postprocessing a uživatelské vlaječky softwaru.

Nastavení parametrů dýchání umožňuje změnit: půlperiodu dechu, počet period, hloubku nádechu a změnu inkrementu interpolace (tj. zrychlené dýchání). Lokace uložení se týká cesty, formátu snímku a počtu kousků, na který lze snímek horizontálně rozdělit při uložení. Parametry animace se týkají změny bodů interpolace a jejich hloubky dýchání. Lze měnit výšku interpolovaných bodů modelů animace (obr. 28).

Součástí nastavení postprocessingu je velikost kernelu gaussovského operátoru (obr. 6). Operátor slouží k rozmazání celé scény a kontury ještěrky. Uživatelské vlaječky umožňují ovlivnit některé části chodu programu: použití více vláken pro generaci textur, využití funkcí implementovaných v Cythonu (načítání modelů, textur), vykreslování do neviditelného okna, ukládání snímků a aplikaci postprocessingu.

Velké množství generovaných snímků má výrazný rozdíl domén způsobený nevhodným nastavením parametrů animace (viz diskuze výsledků 7). Profil deformace skutečných snímků je složitý a na popsání parametry generátoru zdlouhaví, co se týče ručního nastavení uživatelem. Některé syntetické snímky jsou bližší skutečným než jiné. Důvodem je náhodné přidělení hodnot parametrům generátoru, které někdy zvolí vhodné nastavení.

SEZNAM POUŽITÉ LITERATURY

[1] *A Primer on Bézier Curves* [online]. 2013-06-13. [cit. 2023-05-20]. Dostupné z: https://pomax.github.io/bezierinfo.

[2] *Algorithm behind Converting a Diffuse Texture to a Normal Texture?* [online]. NVIDIA Developer Forums, 2022-02-17. [cit. 2023-04-24]. Dostupné z: https://forums.developer.nvidia.com/t/algorithm-behind-converting-a-diffuse-texture-to-a-normal-texture/203220/2.

[3] ALI, Hazrat; MURAD, Shafaq; SHAH, Zubair. Spot the Fake Lungs: Generating Synthetic Medical Images Using Neural Diffusion Models. In: LONGO, Luca; O'REILLY, Ruairi eds. *Artificial Intelligence and Cognitive Science* [online]. Cham: Springer Nature Switzerland, 2023, vol. 1662, s. 32–39 [cit. 2023-03-26]. ISBN 978-3-031-26438-2. Dostupné z DOI: 10.1007/978-3-031-26438-2_3.

[4] ATHITSOS, V.; SCLAROFF, S. Estimating 3D Hand Pose from a Cluttered Image. In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* [online]. Madison, WI, USA: IEEE Comput. Soc, 2003, sv. 2, s. II-432–9 [cit. 2023-03-13]. ISBN 978-0-7695-1900-5. Dostupné z DOI: 10.1109/CVPR.2003.1211500.

[5] BATCHELOR, Bruce G. ed. *Machine Vision Handbook* [online]. London: Springer London, 2012 [cit. 2023-03-07]. ISBN 978-1-84996-169-1. Dostupné z DOI: 10.1007/978-1-84996-169-1.

[6] BERGMANN, Urs; JETCHEV, Nikolay; VOLLGRAF, Roland. Learning Texture Manifolds with the Periodic Spatial GAN [online]. 2017 [cit. 2023-03-24]. Dostupné z DOI: 10.48550/ARXIV.1705.06566.

[7] *Bounding Box Software - Materialize* [online]. [cit. 2023-04-22]. Dostupné z: https://boundingboxsoftware.com/materialize/.

[8] COLL-FONT, Jaume; AFACAN, Onur; HOGE, Scott; GARG, Harsha; SHASHI, Kumar; MARAMI, Bahram; GHOLIPOUR, Ali; CHOW, Jeanne; WARFIELD, Simon; KURUGOL, Sila. Retrospective Distortion and Motion Correction for Free-Breathing DW-MRI of the Kidneys Using Dual-Echo EPI and Slice-to-Volume Registration. *Journal of Magnetic Resonance Imaging* [online]. 2021, vol. 53, no. 5, s. 1432–1443 [cit. 2023-05-04]. ISSN 1053-1807, ISSN 1522-2586. Dostupné z DOI: 10.1002/jmri.27473.

[9] COLL-FONT, Jaume; CHEN, Shi; EDER, Robert; FANG, Yiling; HAN, Qiao Joyce; VAN DEN BOOMEN, Maaike; SOSNOVIK, David E.; MEKKAOUI, Choukri; NGUYEN, Christopher T. Manifold-based Respiratory Phase Estimation Enables Motion and Distortion Correction of Free-breathing Cardiac Diffusion Tensor MRI. *Magnetic Resonance in Medicine* [online]. 2022, vol. 87, no. 1, s. 474–487 [cit. 2023-05-04]. ISSN 0740-3194, ISSN 1522-2594. Dostupné z DOI: 10.1002/mrm.28972.

[10] DEMANT, Christian; STREICHER-ABEL, Bernd; GARNICA, Carsten. *Industrial Image Processing: Visual Quality Control in Manufacturing* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013 [cit. 2023-03-08]. ISBN 978-3-642-33905-9. Dostupné z DOI: 10.1007/978-3-642-33905-9.

[11] DHARIWAL, Prafulla; NICHOL, Alex. Diffusion Models Beat GANs on Image Synthesis [online]. 2021 [cit. 2023-03-24]. Dostupné z DOI: 10.48550/ARXIV.2105.05233.

[12] DOERSCH, Carl. Tutorial on Variational Autoencoders [online]. 2016 [cit. 2023-03-23]. Dostupné z DOI: 10.48550/ARXIV.1606.05908.

[13] EFROS, Alexei A.; FREEMAN, William T. Image Quilting for Texture Synthesis and Transfer. In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* [online]. ACM, 2001, s. 341–346 [cit. 2023-04-22]. ISBN 978-1-58113-374-5. Dostupné z DOI: 10.1145/383259.383296.

[14] FOFONJKA, Anamarija; MILINKOVITCH, Michel C. Reaction-Diffusion in a Growing 3D Domain of Skin Scales Generates a Discrete Cellular Automaton. *Nature Communications* [online]. 2021, vol. 12, no. 1, s. 2433 [cit. 2023-05-24]. ISSN 2041-1723. Dostupné z DOI: 10.1038/s41467-021-22525-1.

[15] FOUNDATION, Blender. *Blender.Org - Home of the Blender Project - Free and Open 3D Creation Software* [online]. blender.org. [cit. 2023-03-27]. Dostupné z: https://www.blender.org/.

[16] GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning.* Cambridge, Massachusetts: The MIT Press, 2016. Adaptive Computation and Machine Learning. ISBN 978-0-262-03561-3.

[17] GUPTA, Ankush; VEDALDI, Andrea; ZISSERMAN, Andrew. Synthetic Data for Text Localisation in Natural Images [online]. 2016 [cit. 2023-04-30]. Dostupné z DOI: 10.48550/ARXIV.1604.06646.

[18] HINTERSTOISSER, Stefan; LEPETIT, Vincent; WOHLHART, Paul; KONOLIGE, Kurt. On Pre-trained Image Features and Synthetic Images for Deep Learning. In: LEAL-TAIXÉ, Laura; ROTH, Stefan eds. *Computer Vision – ECCV 2018 Workshops* [online]. Cham: Springer International Publishing, 2019, vol. 11129, s. 682–697 [cit. 2023-03-13]. ISBN 978-3-030-11009-3. Dostupné z DOI: 10.1007/978-3-030-11009-3_42.

[19] HO, Jonathan; JAIN, Ajay; ABBEEL, Pieter. Denoising Diffusion Probabilistic Models [online]. 2020 [cit. 2023-03-24]. Dostupné z DOI: 10.48550/ARXIV.2006.11239.

[20] HO, Jonathan; SALIMANS, Tim. Classifier-Free Diffusion Guidance [online]. 2022 [cit. 2023-03-24]. Dostupné z DOI: 10.48550/ARXIV.2207.12598.

[21] *In-Sight 9902L Datasheet* [online]. 2020. [cit. 2023-03-07]. DS-IS9902L-10-2020. Cognex. Dostupné z: https://www.cognex.com/downloads/in-sight-9902l-datasheet.

[22] JAMES, Gareth; WITTEN, Daniela; HASTIE, Trevor; TIBSHIRANI, Robert. *An Introduction to Statistical Learning: With Applications in R*. Second edition. New York NY: Springer, 2021. Springer Texts in Statistics. ISBN 978-1-07-161418-1. Dostupné z DOI: 10.1007/978-1-0716-1418-1.

[23] JETCHEV, Nikolay; BERGMANN, Urs; VOLLGRAF, Roland. Texture Synthesis with Spatial Generative Adversarial Networks [online]. 2016 [cit. 2023-03-24]. Dostupné z DOI: 10.48550/ARXIV.1611.08207.

[24] KANEVA, Biliana; TORRALBA, Antonio; FREEMAN, William T. Evaluation of Image Features Using a Photorealistic Virtual World. In: *2011 International Conference on Computer Vision* [online]. Barcelona, Spain: IEEE, 2011, s. 2282–2289 [cit. 2023-03-13]. ISBN 978-1-4577-1101-5. Dostupné z DOI: 10.1109/ICCV.2011.6126508.

[25] KLEIN, G.; MURRAY, D.W. Simulating Low-Cost Cameras for Augmented Reality Compositing. *IEEE Transactions on Visualization and Computer Graphics* [online]. 2010, roč. 16, č. 3, s. 369–380 [cit. 2023-03-26]. ISSN 1077-2626. Dostupné z DOI: 10.1109/TVCG.2009.210.

[26] LE, Quoc V.; RANZATO, Marc'Aurelio; MONGA, Rajat; DEVIN, Matthieu; CHEN, Kai; CORRADO, Greg S.; DEAN, Jeff; NG, Andrew Y. Building High-Level Features Using Large Scale Unsupervised Learning [online]. 2011 [cit. 2023-03-13]. Dostupné z DOI: 10.48550/ARXIV.1112.6209.

[27] *Learn OpenGL, Extensive Tutorial Resource for Learning Modern OpenGL* [online]. [cit. 2023-03-16]. Dostupné z: https://learnopengl.com/.

[28] LEWIS, J.P. Fast Normalized Cross-Correlation. *Ind. Light Magic*. 2001, roč. 10.

[29] *Lizard v1 Free 3D Model - .Obj .Stl - Free3D* [online]. [cit. 2023-03-15]. Dostupné z: https://free3d.com/3d-model/lizard-v1--100686.html.

[30] MARIN, Javier; VAZQUEZ, David; GERONIMO, David; LOPEZ, Antonio M. Learning Appearance in Virtual Scenarios for Pedestrian Detection. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* [online]. San Francisco, CA, USA: IEEE, 2010, s. 137–144 [cit. 2023-03-14]. ISBN 978-1-4244-6984-0. Dostupné z DOI: 10.1109/CVPR.2010.5540218.

[31] MARTIN-BRUALLA, Ricardo; RADWAN, Noha; SAJJADI, Mehdi S. M.; BARRON, Jonathan T.; DOSOVITSKIY, Alexey; DUCKWORTH, Daniel. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections [online]. 2020 [cit. 2023-03-26]. Dostupné z DOI: 10.48550/ARXIV.2008.02268.

[32] MILDENHALL, Ben; SRINIVASAN, Pratul P.; TANCIK, Matthew; BARRON, Jonathan T.; RAMAMOORTHI, Ravi; NG, Ren. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis [online]. 2020 [cit. 2023-03-26]. Dostupné z DOI: 10.48550/ARXIV.2003.08934.

[33] MILL, Leonid; WOLFF, David; GERRITS, Nele; PHILIPP, Patrick; KLING, Lasse; VOLLNHALS, Florian; IGNATENKO, Andrew; JAREMENKO, Christian; HUANG, Yixing; DE CASTRO, Olivier; AUDINOT, Jean-Nicolas; NELISSEN, Inge; WIRTZ, Tom; MAIER, Andreas; CHRISTIANSEN, Silke. Synthetic Image Rendering Solves Annotation Problem in Deep Learning Nanoparticle Segmentation. *Small Methods* [online]. 2021, vol. 5, no. 7, s. 2100223 [cit. 2023-03-07]. ISSN 2366-9608. Dostupné z DOI: 10.1002/smtd.202100223.

[34] NGUYEN, Loc X.; SONE AUNG, Pyae; LE, Huy Q.; PARK, Seong-Bae; HONG, Choong Seon. A New Chapter for Medical Image Generation: The Stable Diffusion Method. In: *2023 International Conference on Information Networking (ICOIN)* [online]. Bangkok, Thailand: IEEE, 2023, s. 483–486 [cit. 2023-03-26]. ISBN 978-1-66546-268-6. Dostupné z DOI: 10.1109/ICOIN56518.2023.10049010.

[35] *OpenCV: Sobel Derivatives* [online]. [cit. 2023-04-25]. Dostupné z: https://docs.opencv.org/3.4/d2/d2c/tutorial_sobel_derivatives.html.

[36] OPHUS, Colin; CISTON, Jim; NELSON, Chris T. Correcting Nonlinear Drift Distortion of Scanning Probe and Scanning Transmission Electron Microscopies from Image Pairs with Orthogonal Scan Directions. *Ultramicroscopy* [online]. 2016, vol. 162, s. 1–9 [cit. 2023-05-04]. ISSN 03043991. Dostupné z DOI: 10.1016/j.ultramic.2015.12.002.

[37] RAAD, Lara; GALERNE, Bruno. Efros and Freeman Image Quilting Algorithm for Texture Synthesis. *Image Processing On Line* [online]. 2017, roč. 7, s. 1–22 [cit. 2023-04-22]. ISSN 2105-1232. Dostupné z DOI: 10.5201/ipol.2017.171.

[38] RAJPURA, Param S.; BOJINOV, Hristo; HEGDE, Ravi S. Object Detection Using Deep CNNs Trained on Synthetic Images [online]. 2017 [cit. 2023-03-14]. Dostupné z DOI: 10.48550/ARXIV.1706.06782.

[39] REN, Zhongzheng; LEE, Yong Jae. Cross-Domain Self-supervised Multi-task Feature Learning Using Synthetic Imagery [online]. 2017 [cit. 2023-03-24]. Dostupné z DOI: 10.48550/ARXIV.1711.09082.

[40] ROZANTSEV, Artem; LEPETIT, Vincent; FUA, Pascal. On Rendering Synthetic Images for Training an Object Detector. *Computer Vision and Image Understanding* [online]. 2015, vol. 137, s. 24–37 [cit. 2023-03-07]. ISSN 10773142. Dostupné z DOI: 10.1016/j.cviu.2014.12.006.

[41] *Search Media - Wikimedia Commons* [online]. [cit. 2023-03-15]. Dostupné z: https://commons.wikimedia.org/w/index.php?go=Go&search=Lacertae+skin&title=Special%3AMediaSearch.

[42] SHRIVASTAVA, Ashish; PFISTER, Tomas; TUZEL, Oncel; SUSSKIND, Josh; WANG, Wenda; WEBB, Russ. Learning from Simulated and Unsupervised Images through Adversarial Training [online]. 2016 [cit. 2023-03-26]. Dostupné z DOI: 10.48550/ARXIV.1612.07828.

[43] *SmartView Liniová Světla Datasheet* [online]. 2022. [cit. 2023-03-08]. Line-Light-LS-60-LS-200-LS-400. SmartView. Dostupné z: https://www.smartview.cz/p-liniova-svetla.

[44] SONG, Liangchen; XU, Yonghao; ZHANG, Lefei; DU, Bo; ZHANG, Qian; WANG, Xinggang. Learning From Synthetic Images via Active Pseudo-Labeling. *IEEE Transactions on Image Processing* [online]. 2020, roč. 29, s. 6452–6465 [cit. 2023-03-24]. ISSN 1057-7149, ISSN 1941-0042. Dostupné z DOI: 10.1109/TIP.2020.2989100.

[45] *Stable Diffusion Prompt Book | OpenArt* [online]. [cit. 2023-03-24]. Dostupné z: https://openart.ai/promptbook.

[46] TEWARI, A.; FRIED, O.; THIES, J.; SITZMANN, V.; LOMBARDI, S.; SUNKAVALLI, K.; MARTIN-BRUALLA, R.; SIMON, T.; SARAGIH, J.; NIESSNER, M.; PANDEY, R.; FANELLO, S.; WETZSTEIN, G.; ZHU, J.-Y.; THEOBALT, C.; AGRAWALA, M.; SHECHTMAN, E.; GOLDMAN, D. B; ZOLLHÖFER, M. State of the Art on Neural Rendering. *Computer Graphics Forum* [online]. 2020, vol. 39, no. 2, s. 701–727 [cit. 2023-03-26]. ISSN 0167-7055, ISSN 1467-8659. Dostupné z DOI: 10.1111/cgf.14022.

[47] TEWARI, Ayush; THIES, Justus; MILDENHALL, Ben; SRINIVASAN, Pratul; TRETSCHK, Edgar; WANG, Yifan; LASSNER, Christoph; SITZMANN, Vincent; MARTIN-BRUALLA, Ricardo; LOMBARDI, Stephen; SIMON, Tomas; THEOBALT, Christian; NIESSNER, Matthias; BARRON, Jonathan T.; WETZSTEIN, Gordon; ZOLLHOEFER, Michael; GOLYANIK, Vladislav. Advances in Neural Rendering [online]. 2021 [cit. 2023-03-27]. Dostupné z DOI: 10.48550/ARXIV.2111.05849.

[48] TOBIN, Josh; FONG, Rachel; RAY, Alex; SCHNEIDER, Jonas; ZAREMBA, Wojciech; ABBEEL, Pieter. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World [online]. 2017 [cit. 2023-03-13]. Dostupné z DOI: 10.48550/ARXIV.1703.06907.

[49] VARGA, T.; BUNKE, H. Generation of Synthetic Training Data for an HMM-based Handwriting Recognition System. In: *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.* [online]. Edinburgh, UK: IEEE Comput. Soc, 2003, sv. 1, s. 618–622 [cit. 2023-03-11]. ISBN 978-0-7695-1960-9. Dostupné z DOI: 10.1109/ICDAR.2003.1227736.

[50] XIEYANG. *Bigjpg - AI Super-Resolution Lossless Image Enlarging / Upscaling Tool Using Deep Convolutional Neural Networks* [online]. [cit. 2023-03-15]. Dostupné z: https://bigjpg.com.

[51] *XROMM: Ventilatory Rib Kinematics in the Savannah Monitor Lizard* [online]. [cit. 2023-03-15]. Dostupné z: https://www.xromm.org/projects/varanus-breathing/.

SEZNAM ZKRATEK A SYMBOLŮ

A	Amplituda.	49
b	Hodnota modré složky.	36
$B_{nádech}$	Parametr maximálního nádechu, pozitivní nebo negativní.	50
$B_{výdech}$	Parametr maximálního výdechu, pozitivní nebo negativní.	50
c	Barva.	24, 40, 41
c_{norm}	Barva normalizovaná mezi 0 a 2.55.	41
D	Obraz vzdálenosti na druhou.	35
d_X	Diskriminátor GAN.	22
d_x	Posun v ose x.	49
d_{min}	Minimální hodnota v obrazu vzdálenosti.	35
D_s	Úhly paprsku NeRF.	24
e	Rozdíl vstupního a výstupního vzorku.	37
E_h	Horizontální kumulativní chyba.	37
ε	Tolerance.	34, 35
ε_N	Výběr proměnné z Gaussovského rozdělení.	22
E_v	Vertikální kumulativní chyba.	37
f	Frekvence.	49
fbo	Frame buffer object, OpenGL.	51, 52
F_Θ	Naučená funkce neuronové sítě.	24
$f_{v,vt,vn}$	Stěna modelu.	45
g	Hodnota zelené složky.	36
$G(X)$	Generovaný snímek.	22
γ	Cesta, řez v oblasti překrytí.	36, 37
g_X	Generátor GAN.	22
$glfw$	Viditelné okno z nástrojů pro OpenGL.	51, 52
h	Hodnota ve výškové mapě.	36
I	Jednotková matice.	21
I_0	Vstupní textura.	34–36, 69
I_c	Barevná textura.	40, 41
I_e	Prázdná textura.	40, 41
I_g	Černobílá textura.	40, 41
I_s	Syntetizovaná textura.	34–36, 69
M	Matice prošití.	35, 36
m	Souřadnice řádku pixelu.	34, 35, 40, 41
M_c	Barevná maska.	40, 41
μ	Střední hodnota.	22

M_x	Počet řádků.	40
M_y	Počet sloupců.	15
N	Gaussovské rozdělení.	21
n	Souřadnice sloupce pixelu.	34, 35, 40, 41
$\bar{n}$	Normalizovaná normála.	39
n_g	Počet kandidátů z barevné masky.	40, 41
y_{norm}	Normalizovaná souřadnice y.	49
N_x	Počet řádků.	15
n_x	X, složka normály.	38, 45
N_y	Počet sloupců.	40
n_y	Y, složka normály.	38, 45
n_z	Z, složka normály.	38, 45
o	Offset.	49
P	Množina pixelů.	40, 41
p	Pixel.	40, 41
$P(X)$	Rozdělení pravděpodobnosti v obrazu.	21
P_{ext}	Vzorek o velikosti vstupní textury.	35, 36
ϕ	Fázový posun.	49
ϕ_d	Azimut úhel paprsku NeRF.	24
P_{in}	Vzorek vstupní textury.	34–37, 69
P_{new}	Nový vzorek syntetizované textury.	34, 35
P_{old}	Vzorek syntetizované textury.	34–37
p_θ	Dekodér VAE.	21
Q	Binární matice překrytí.	35, 36
Q_{ext}	Binární matice o velikosti vstupní textury.	35, 36
q_θ	Enkodér VAE.	21
R	Poměr výstup/vstup.	34
r	Hodnota červené složky.	36
$\bar{R}$	Normalizovaný vektor odrazu.	39
ρ	Hustota.	24
r_o	Rádius okolí pixelu.	40
$\bar{S}$	Normalizovaný směr osvětlení.	39
s	Intenzita osvětlení.	39
σ	Směrodatná odchylka.	22
S_x	Měřítko v ose x.	49
S_y	Měřítko v ose y.	49
S_z	Měřítko v ose z.	49
T	Počet kroků difuze.	23
t	Čas.	23

t_1	Počátek paprsku NeRF.	24
t_2	Konec paprsku NeRF.	24
T_h	Matice horizontální zpětné cesty.	36, 37
Θ	Polární úhel od normály.	39
θ_d	Polární úhel paprsku NeRF.	24
t_i	Parametr Bézierovy křivky.	33, 51
T_v	Matice vertikální zpětné cesty.	36, 37
T_x	Translace v ose x.	49
T_y	Translace v ose y.	49
T_z	Translace v ose z.	49
v	Souřadnice bodu modelu.	45
v_{gd}	Ocenění GAN.	22
vn	Souřadnice normál modelu.	45
vt	Souřadnice textury modelu.	45
w_o	Velikost překrytí.	34, 35, 37
w_p	Velikost vzorku.	34, 35, 37
X	Obraz, snímek.	21, 22
X'	Výstupní obraz.	22
x_0	Počáteční stav difuze.	23
x_{curr}	Aktuální číslo snímku.	33, 50, 51
x_{new}	Následující číslo snímku.	33, 50, 51
x_{old}	Předcházející číslo snímku.	33, 50, 51
$X_{x,y,z}$	Souřadnice bodu v prostoru NeRF.	24
y_0	První model ještěrky, bod interpolace.	33, 50
y_1	Druhý model ještěrky, bod interpolace.	33, 50
y_2	Třetí model ještěrky, bod interpolace.	33
y_3	Čtvrtý model ještěrky, bod interpolace.	33
y_{curr}	Aktuální model ještěrky, bod interpolace.	33, 50
Z	Latentní, skrytý prostor neuronové sítě.	21
Z_x	Proměnná v latentním prostoru.	21, 22

SEZNAM OBRÁZKŮ

1 Uspořádání plošného senzoru, Interline 15
2 Uspořádání řádkového senzoru s řádkovým zásobníkem 16

3 Transformace textu ze zdroje s malou variací 18
4 Prostředí v Blenderu 19
5 Obecná posloupnost operací při vykreslování 20
6 Příklad masky 5×5 aproximující Gaussovské rozložení. 20
7 Obecná struktura VAE 22
8 Obecná struktura GAN 23
9 Příklad řetězců difuze 23
10 Výsledek difuze, podnět: Common lizard viewed from above. 24
11 NeRF Raymarching 25

12 Skutečný výstup řádkové kamery, ještěrka obecná 29
13 Model ještěrky 30

14 Možný výběr počátku vzorku P_{in} v I_0, (zeleně) 35
15 Možné tvary překrytí vzorků v I_s 36
16 Sobelův operátor, kernel 3×3 38
17 Scharrův operátor, kernel 3×3 38
18 Směrované osvětlení. 39

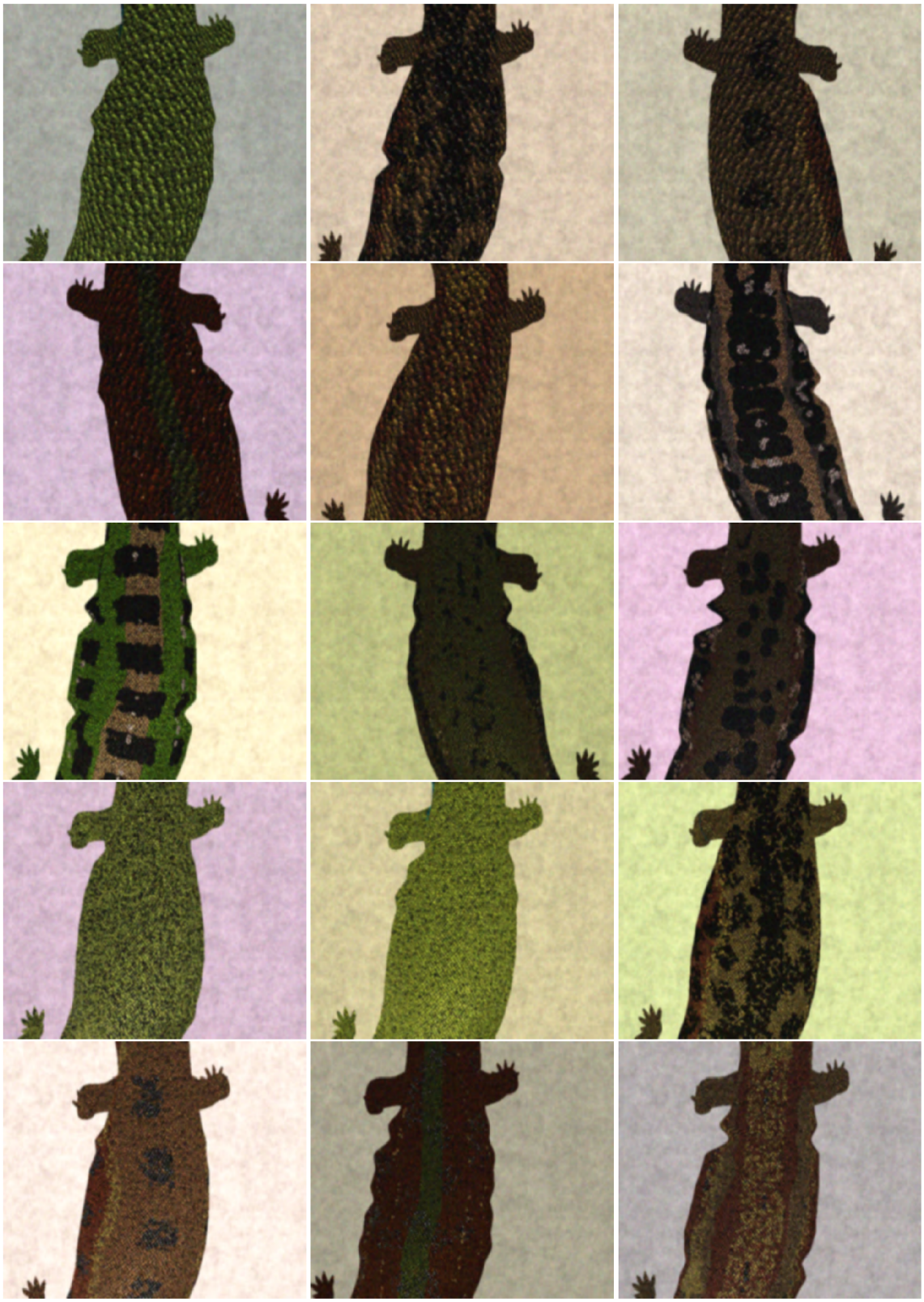
19 Schéma implementovaného generátoru dat. 43
20 Souřadnicový systém Blenderu, OpenGL a pohledu shora 44
21 Obecná posloupnost operací generátoru textur 46
22 Příklad nově generované masky, albedo, normálová a zvýraznění (zleva) 47
23 Výsledek násobení barevné masky a textury šupin po vykreslení 48
24 Okolní, difuzní a komponenta odlesků Phong modelu. 48
25 Matice projekce, změna měřítka a translace 49
26 Nastavení rozlišení a parametrů dýchání. 50
27 Nastavení datové sady, uložení a spuštěni. 51
28 Interpolace modelu v generátoru. 52
29 Generovaná deformace, postprocessing a skutečný snímek. 52
30 Způsob označení snímku. 53

31 Reálné snímky pořízené řádkovou kamerou 79
32 Syntetické snímky z generátoru dat 80

SEZNAM PŘÍLOH

A Příklady variací výstupu **73**
B Odkaz na software **77**
C Reálné a syntetické snímky **79**

A Příklady variací výstupu

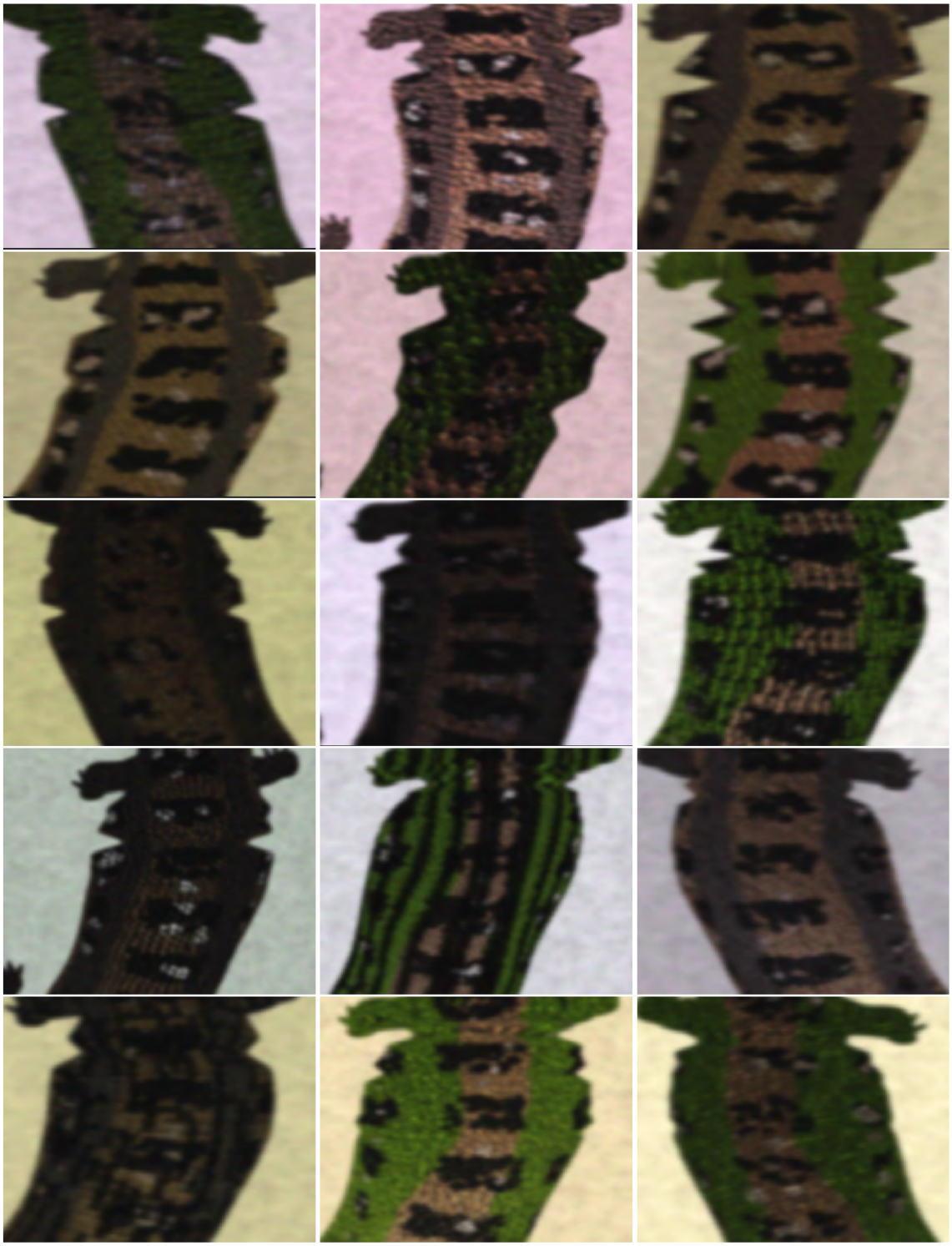


B Odkaz na software

Všechny soubory a verze softwaru generátoru jsou dostupné z: `https://github.com/PFurik/DP_2023.git`

C Reálné a syntetické snímky

Obr. 31: Reálné snímky pořízené řádkovou kamerou



Obr. 32: Syntetické snímky z generátoru dat