PROGRAM PRO PODPORU AUTOMATICKÉ DETEKCE KMENE STROMŮ V SADU

Martin Vitoušek [1]***, Matouš Cejnek*** [1]***, Jakub Jura*** [1]***, Pavel Trnka*** [1]***, Lubor Zelený***[2]

[1] ***Department of Instrumentation and Control, Czech Technical University in Prague, Technicka 4, Prague 6, martin.vitousek@fs.cvut.cz***

[2] ***Research and Breeding Institute of Pomology Holovousy, Holovousy 129, 50801, lubor.zeleny@vsuo.cz***

Abstrakt: Tento článek popisuje vývoj softwarového nástroje pro detekci kmenů stromů na obrazových snímcích s důrazem na přesnost a konzistenci výsledků bez ohledu na roční období. Program je implementován v jazyce Python a využívá technologie jako NumPy, OpenCV a onnxruntime. Detekční model je založen na architektuře YOLOv8 a natrénován na datech z jablečných sadů. Program je schopen identifikovat kmeny stromů do dvou metrů od kamery a spolehlivě rozlišuje mezi kmeny a prvky vodící konstrukce. Tento nástroj nabízí efektivní a flexibilní řešení pro aplikace vyžadující přesnou analýzu obrazových dat.

Klíčová slova: strojové vidění, deep learning, kmeny stromů, detekce kmenů, automatizace sadu, precizní zemědělství

Abstract: This article describes the development of a software tool for detecting tree trunks in images, focusing on accuracy and consistency of results regardless of the season. The program is implemented in Python and utilizes technologies such as NumPy, OpenCV, and onnxruntime. The detection model is based on the YOLOv8 architecture and trained on data from apple orchards. The program can identify tree trunks within two meters of the camera and reliably distinguish between trunks and structural elements. This tool offers an efficient and flexible solution for applications requiring precise image data analysis.

Keywords: machine vision, deep learning, tree trunk, trunk detection, orchard automation, Precision agriculture

1 Úvod

V současné době dochází k rychlému rozvoji nasazení systémů umělé inteligence a robotiky v zemědělství a zvláště potom v precizním zemědělství. Tento článek se zaměřuje na oblast ovocnářství a nové koncepce sadů s využitím technologií Agriculture 4.0. Jednou z relativně často nasazovaných technologií je autonomní robotika určená pro monitoring, ošetřování a sklízení ovocných stromů. Základní úlohou autonomního robota je v tomto případě orientace v sadu a pro využití v precizním zemědělství i identifikace jednotlivých stromů. Identifikace kmene strumu je proto pro navigaci robota zásadní. Detekované kmeny jsou často využívány právě k navigaci autonomních robotů [1]. V dnešní době již klasickými metodami pro navigaci robota je využití dat z lidaru případně ultrazvukových senzorů. V obou případech lze z dat identifikovat kandidáty na kmeny. Vlivem rozvoje metod tzv. hlubokého učení došlo v posledních letech k využití detekce kmenů z obrazových dat a lidaru [2]. V jiné studii bylo navrženo používat jak běžné RGB snímky, tak termální snímky [3], snímky v NIR spektru, či hloubkové kamery. Detekce kmenů je ve větším měřítku řešena při pěstování vinné révy [4], [5] se zabývají srovnáním YOLOv3 a MobileNet architektury pro detekci kmenů, přičemž využití je zamýšlené právě pro navigaci mobilního robota. Gao a kolektiv [6] využívají detekci kmenů jako pomocnou při detekci, respektive počítání množství plodů na stromě. V tomto článku představujeme software sloužící detektor jednotlivých kmenů z RGB snímků, který je použitelný i pro uživatele neznalé metod hlubokého učení a pokročilých nástrojů využívaných ve studiích výše.

2 Funkce

Program je navržen tak, aby analyzoval obrazové snímky s cílem identifikovat kmeny stromů a přesně určit jejich pixelovou pozici. Té může být následně využito například systémem navigace robota v sadu nebo systémem získávání snímků stromů z kontinuálního kamerového záznamu. Hlavním úkolem programu je detekovat pouze ty části stromového kmene, které nejsou zakryty korunou, což zajišťuje konzistentní výsledky nezávislé na ročním období, a to i v případě, kdy jsou stromy olistěné. Tento přístup je klíčový pro dosažení spolehlivých výsledků při různých podmínkách nasnímání.

Program je dále optimalizován pro detekci stromů v určité vzdálenosti od kamery, konkrétně do dvou metrů při použití širokoúhlého objektivu f = 3.5 mm při velikosti snímače 4.54 x 3.42 mm. RSPPX kamery je 0.76 mm, tedy s úhlem záběru 66°. Stromy nacházející se mimo tuto definovanou vzdálenost od objektivu jsou považovány za pozadí a jsou programem při detekci ignorovány. Tím se minimalizuje riziko falešných pozitiv a zvyšuje se přesnost detekce kmenů, které jsou relevantní pro daný úkol.

Další klíčovou vlastností programu je jeho schopnost bezpečně odlišit stromy od různých prvků vodící konstrukce, které mohou být přítomny na snímcích. Program je navržen tak, aby tyto prvky spolehlivě rozpoznal a nezaměňoval je za kmeny stromů, čímž se dále zvyšuje přesnost a spolehlivost detekce.

3 Implementace

Tento program byl vyvinut v programovacím jazyce Python, který je zároveň nezbytný jako runtime prostředí pro jeho provoz. Lze jej využít jak jako samostatnou aplikaci, tak i jako knihovnu, kterou lze snadno integrovat do jakéhokoli Pythonového projektu.

Základní struktura programu je postavena na několika klíčových nástrojích, které zajišťují vysokou výpočetní efektivitu a spolehlivost. Mezi hlavní komponenty patří NumPy [7] pro pokročilé matematické výpočty, OpenCV [8] pro práci s obrazovými daty, a onnxruntime pro implementaci modelů hlubokého učení. Tyto nástroje jsou považovány za standardy ve svých oblastech – NumPy se hojně využívá pro numerické výpočty, OpenCV je běžně používáno pro zpracování a analýzu obrazů, a onnxruntime poskytuje výkonné prostředí pro provoz neuronových sítí ve formátu Open Neural Network Exchange (ONNX).

Model hlubokého učení, který je jádrem programu, je založen na moderní a populární architektuře YOLOv8 [9] ve verzi nano. Tento model byl natrénován na datové sadě získané v jablečných sadech, kde byla data ručně anotována pro účely tréninku a validace. Po dokončení tréninku byl model exportován do formátu ONNX, což zajišťuje jeho snadnou integraci a použití v navržené aplikaci. Díky onnxruntime může být model provozován bez potřeby dalších závislostí, což výrazně usnadňuje jeho nasazení. V případě potřeby je možné model snadno nahradit jiným ONNX modelem, čímž se program stává flexibilním a přizpůsobitelným pro různé aplikace a scénáře.

4 Instalace a použití

Program je navržen tak, aby byl plně kompatibilní s operačními systémy Windows i Linux, což zajišťuje jeho širokou dostupnost a flexibilitu při nasazení v různých pracovních prostředích. Tento univerzální přístup znamená, že uživatelé mohou využívat program na svém preferovaném operačním systému bez nutnosti volit specifické platformy.

Instalace samotného programu není nutná, což výrazně usnadňuje jeho použití a nasazení. Namísto tradiční instalace je však třeba připravit vhodné prostředí pro běh programu, které zahrnuje instalaci Pythonu, což je základní runtime prostředí potřebné pro provoz aplikace. Dále je pak nutné nainstalovat veškeré závislosti odkázané v souboru requirements.txt, který je součástí balíčku. Toto prostředí může být vytvořeno několika způsoby, v závislosti na preferencích a potřebách uživatele.

Uživatelé mohou zvolit systémové prostředí a instalovat všechny závislosti globálně. Tento přístup je vhodný pro uživatele, kteří chtějí program provozovat na stabilní a trvalé konfiguraci svého systému a současně je jisté, že nebudou paralelně pracovat s dalšími projekty na stejném zařízení.

Standardní a doporučený postup při práci s Python projekty je však využití virtuálního prostředí, což je izolované prostředí specifické pro daný projekt nebo aplikaci. Virtuální prostředí umožňuje instalaci všech potřebných knihoven a závislostí bez ovlivnění zbytku systému. Tento přístup je často preferován vývojáři a uživateli, kteří pracují s více projekty najednou, nebo potřebují udržovat různé verze knihoven a balíčků pro různé aplikace. Virtuální prostředí tedy poskytuje větší flexibilitu a kontrolu nad nastavením a závislostmi programu.

Jednoduché použití představeného programu jako knihovny pro libovolný obrázek ve formátu *NumPy array* je demonstrován následovně:

```
from detector import Detector
detector = Detector()
findings = detector.detect(image)
```

kde *findings* je seznam detekcí kmenů v daném snímku. Přes jednotlivé detekce je možné iterovat například následovně:

```
for box, score, cls_id in findings:
    # do something
```

kde *box* je NumPy matice představující ohraničující krabici (x, y, h, w) a *score* je pravděpodobnost, že je to skutečně kmen. Složitější příklad, ve kterém je představený software využit k anotování ohraničujících boxů kmenů přímo do snímků následuje:

```
import cv2 as cv
from src.detector import Detector
detector = Detector()
image = cv.imread("example_data/input1.jpg")
findings = detector.detect(image)
for box, score, cls_id in findings:
    cv.rectangle(image, tuple(box[:2]), tuple(box[2:]), (255, 255, 0), 2)
cv.imwrite("example_data/output1.jpg", image)
cv.imshow("Findings", image)
cv.waitKey(10000)
```

Výstup tohoto příkladu pro ukázkový snímek je představen na Obr. 1.

5 Závěr

Závěrem lze říci, že vyvinutý softwarový nástroj úspěšně řeší problém přesné detekce kmenů stromů na obrazových snímcích a poskytuje konzistentní výsledky napříč různými ročními obdobími. Díky využití pokročilých technologií, jako jsou NumPy, OpenCV a onnxruntime, dosahuje program vysoké výpočetní efektivity a přesnosti při detekci. Integrace modelu inspirovaného YOLOv8, který byl natrénován na pečlivě anotovaném datasetu z jablečných sadů, zajišťuje spolehlivou identifikaci kmenů stromů do vzdálenosti dvou metrů a zároveň efektivně rozlišuje mezi kmeny a jinými prvky konstrukce. Schopnost programu fungovat na platformách Windows i Linux bez potřeby složité instalace zvyšuje jeho uživatelskou přívětivost a dostupnost. Tento software nabízí robustní a flexibilní řešení pro různé aplikace v zemědělství a příbuzných oblastech, kde je klíčová přesná analýza obrazových dat.

Software představený v tomto článku byl otestován pro identifikaci jednotlivých stromů z kontinuálního kamerového záznamu) kde jsou výstupy detektoru fúzovány s daty z lidaru a RTK-GPS. Dalším využitím je real-time navigace robota při průjezdu sadem za účelem dalšího monitoringu (například počítání plodů nebo květenství).

Poděkování

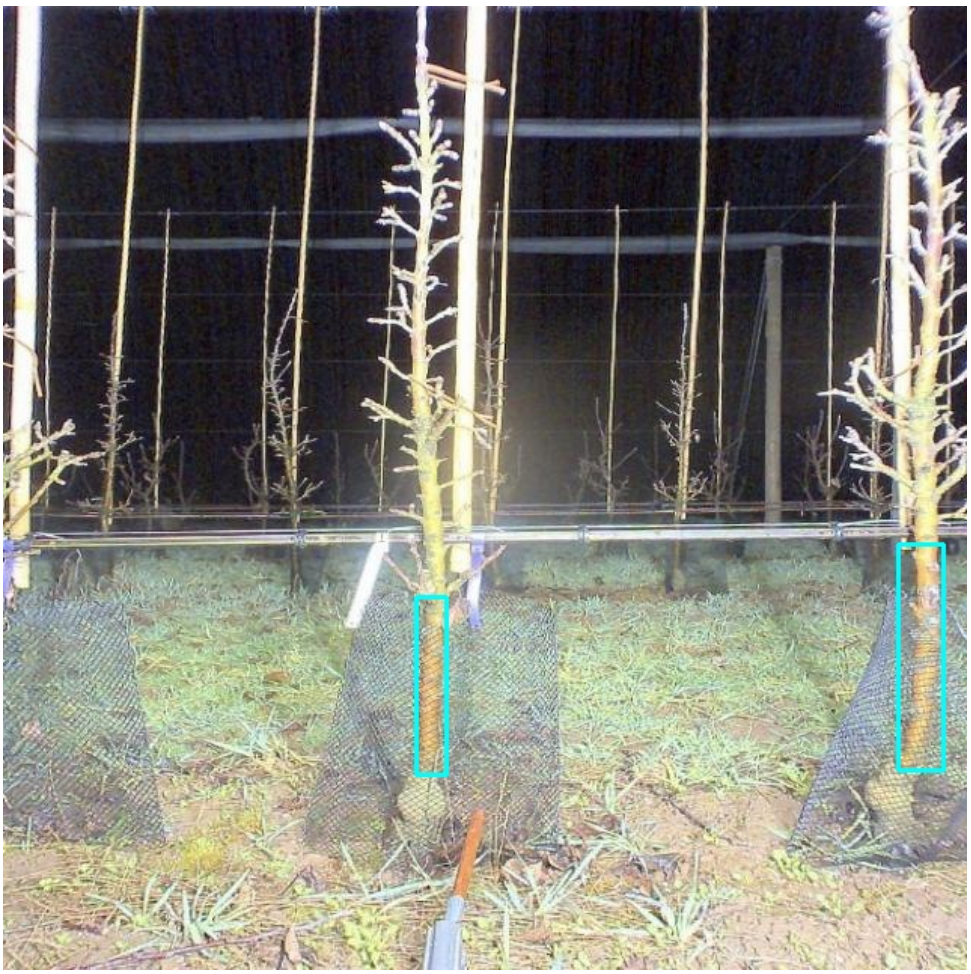
Software a článek byl podpořen projektem QK21010170 Nová koncepce sadů s nástupem technologií 4.0.

Literatura

[1] Peichen Huang, Peikui Huang, Zihong Wang, Xiao Wu, Jie Liu, and Lixue Zhu. Deep-learning-based trunk perception with depth estimation and dwa for robust navigation of robotics in orchards. *AGRONOMY-BASEL*, 13(4), APR 2023.

[2] Hiroki Kurita, Masaki Oku, Takeshi Nakamura, Takeshi Yoshida, and Takanori Fukao. Localization method using camera and lidar and its application to autonomous mowing in orchards. *JOURNAL OF ROBOTICS AND MECHATRONICS*, 34(4, SI):877–886, AUG 2022.

[3] Ailian Jiang, Ryozo Noguchi, and Tofael Ahamed. Tree trunk recognition in orchard autonomous operations under different light conditions using a thermal camera and faster r-cnn. *SENSORS*, 22(5), MAR 2022.



Obr. 1: Obrázek snímku automaticky anotovaného (predikovaného) pomocí představeného software. Ohraničující krabice označují detekované kmeny. Za povšimnutí stojí, že na tomto snímku jsou koruny stromů (v zimním období) vizuálně podobně kmenům a model je přesto odliší.

[4] Andre Silva Aguiar, Filipe Neves Dos Santos, Armando Jorge Miranda De Sousa, Paulo Moura Oliveira, and Luis Carlos Santos. Visual trunk detection using transfer learning and a deep learning-based coprocessor. *IEEE ACCESS*, 8:77308–77320, 2020.

[5] Andre Silva Pinto de Aguiar, Filipe Baptista Neves dos Santos, Luis Carlos Feliz dos Santos, Vitor Manuel de Jesus Filipe, and Armando Jorge Miranda de Sousa. Vineyard trunk detection using deep learning - an experimental device benchmark. *COMPUTERS AND ELECTRONICS IN AGRICULTURE*, 175, AUG 2020.

[6] Fangfang Gao, Wentai Fang, Xiaoming Sun, Zhenchao Wu, Guanao Zhao, Guo Li, Rui Li, Longsheng Fu, and Qin Zhang. A novel apple fruit detection and counting methodology based on deep learning and trunk tracking in modern orchard. *COMPUTERS AND ELECTRONICS IN AGRICULTURE*, 197, JUN 2022.

[7] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

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

[9] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.