NAO – Knihovník

Zpráva k semestrálnímu projektu

Daniil Myronov

Martin Škopek

Abstrakt

Cílem semestrálního projektu bylo vyvinout klient – server aplikaci pro humanoidního robota NAO, který bude sloužit jako asistent při vyhledávání správné krabice (poličky) pro danou knihu z dynamického katalogu knih.

Obsah

Řešení

Původní myšlenka těsně po výběru témata semestrálního projektu byla autonomní robot procházející se prostorem, ke kterému by bylo možné přijít, předat mu knihu a robot by ji založil do správné krabice. To se vzhledem k nosnosti NAO robota ukázalo jako neproveditelné, spokojili jseme se tedy pouze s klasifikací knihy a lokaci správné krabice.

Prvotní návrh klasifikátoru počítal se statickým indexem známých knih a jejich klasifikací pouze na základě fotografie, bez zpracování textu. Podobně jako např. https://cs229.stanford.edu/proj2015/127_report.pdf.

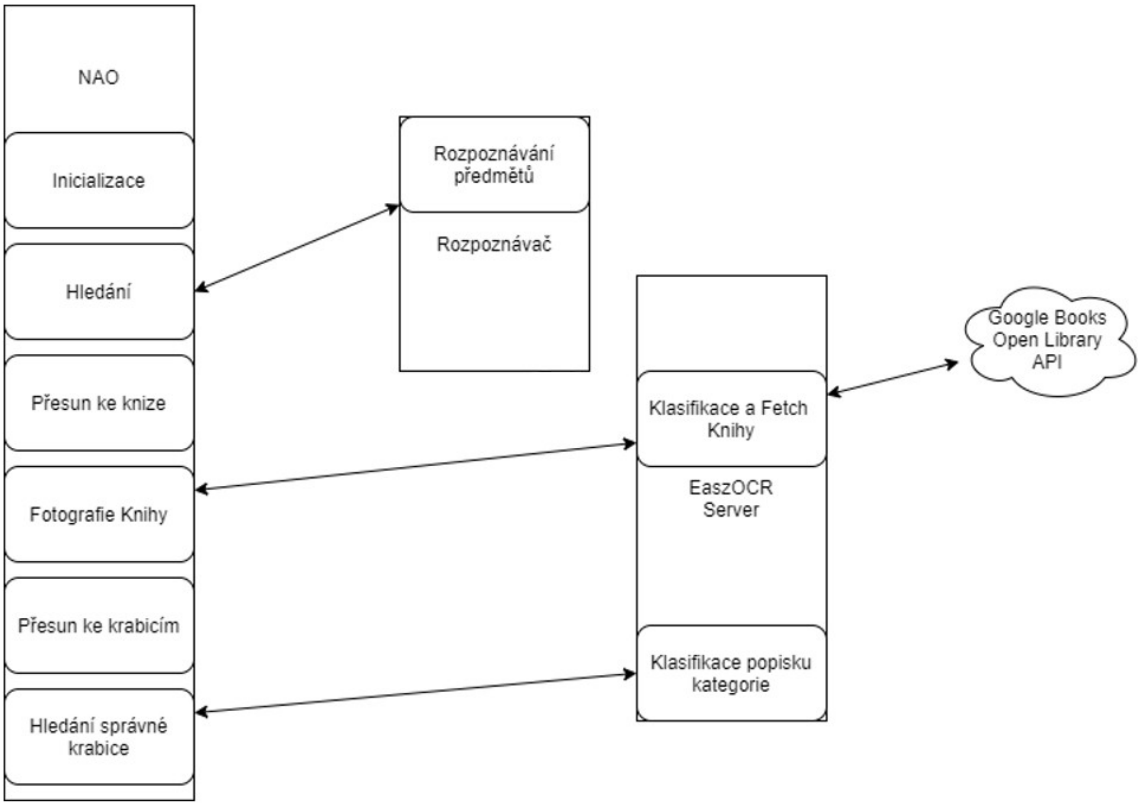
Tato varianta se ukázala jako slepá ulička, zejména kvůli potenciálním nárokům na výkon a kolísavou kvalitou kamer NAO robota.

Nakonec jsme se rozhodli využit textu na obálce a pomocí předtrénovaného klasifikátoru easyOCR (https://github.com/JaidedAI/EasyOCR) se pokusit získat název knihy. V této fázi jsme se rozhodli namísto statického indexu využít služeb veřejně dostupných API Google Books (books.google.com) to se osvědčilo jako dobrý zdroj ISBN, na druhou stranu má problém s poskytováním relevantních kategorii, a proto jsme se rozhodli získané ISBN využít k dotazu vůči API Open Library (https://openlibrary.org/) a tato data doplnit a poté už jen najít správnou krabici.

Klient server architektura

„Klient-server je síťová architektura, která odděluje klienta (často aplikaci s grafickým uživatelským rozhraním) a server, kteří spolu komunikují přes počítačovou síť. Klient–server aplikace obsahují jak klienta, tak i server. Alternativou architektury klient–server je peer-to-peer, kde spolu však komunikují počítače mezi sebou bez serveru. „ (Klient–server, https://cs.wikipedia.org/w/index.php?title=Klient%E2%80%93server (naposledy navštíveno 23. 01. 2023).)

Vzhledem k výpočetní náročnosti rozpoznávání textu jsme se rozhodli využit NAO robota pouze jako agenta prohledávajícího prostředí a samotné rozpoznávání knih a popisků kategorii na krabicích je delegováno na server, se kterým robot komunikuje pomocí REST API



Obrázek 1: Schema projektu (via app.diagrams.net)

Klient

Script spouštěný na klientovi má za cíl ho nechat pohybavat prostorem a v případě, že narazí na knihu, tak jí nasměrovat ke správné krabici. To se dá rozdělit na několik fází:

1. Inicializace

Soubor `run.py`

Uživatel spustí program prostřednictvím souboru run.py a v parametrech zadá informace o robotovi (jméno nebo adresu) a adresy serverů. Nejprve se inicializuje samotná třída robota, zapíše se IP serverů a připravíme si objektové abstrakce API robota.

2. Čekáme na dotyk hlavy

Soubor `nao_librarian.py`, *metody* `wait_for_starting_touch`, `on_starting_touch`

K reakci na dotyk na hlavě slouží servis ALTouch. Podobný algoritmus najdete v dokumentaci: http://doc.aldebaran.com/2-1/naoqi/sensors/altouch.html

3. Hledání

Metoda `look_for_book`

Robot se postupně podívá před sebe 30, respektivě 60 stupňů doleva a pak zkusí to znovu. Pokud knihu stále nenajde, posune se.
Kniha v obraze fotoaparátu se vyhledává pomocí vzdáleného serveru, který používá službu *ALIMDetection*.
Robot prochází prostorem s metodou `look_for_book`, průběžně fotí spodní kamerou a za pomoci metody `find_book` se průběžně dotazuje rozpoznávacího serveru, jestli našel knihu.

4. Kniha nalezena

Metody `get_distance`, `go_to_book`, `move_to_book`, `move_with_stops`

Pomocí metody move_with_stops se iterativně přiblížíme ke knize, abychom měli lepší záběr kamery. Pokud je robot dostatečně blízko knihy, nedělá žádné další zastávky. Vzdálenost k robotovi se vypočítá pomocí metody `get_distance` ze souboru `tools.py`. Vezme se střed knihy na obrázku a pomocí triangulace se určí vzdálenost.

5. Vylepšíme úhel kamery

Metoda `change_posture`

Když se robot přiblíží ke knize, změnou polohy a úhlů kloubů se dostane do správné polohy, aby mohl knihu vyfotografovat. Používané služby jsou *ALRobotPosture*, *ALMotion*.

6. Vyfotíme obálku a provedeme transformaci

Metody `take_book_photo`, `get_warped_image` *z souboru* `perspective_warp.py`

Metoda `get_warped_image` používá algoritmy transformace fotografií a snaží se z obrázku vyříznout pouze část s knihou. K tomuto účelu se používá algoritmus detekce hran Canny. Pokud jsou nalezeny přesně 4 hrany, spočítají se jejich průsečíky a změní se perspektiva obrazu. Algoritmus pro změnu perspektivy je převzat z https://pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/ .

6. Obálku si necháme vyhodnotit serverem

Metody `send_photo_to_server`, `decorate_sending_photo`

Robot odešle obrázek obálky knihy na server pomocí požadavku POST na podadresu */cover*. Protože tento proces může být dlouhý, zatímco robot čeká na odpověď na požadavek, asynchronně se opakují fráze jako "Hm...", "Myslím, že to je..." apod.

7. Zpracování odpovědi serveru

Metody `send_photo_to_server`, `on_book_info_not_found`, `say_book_info`

Pokud návratový kód serveru není 200, spustí se metoda on_book_info_not_found a robot se vrátí na původní pozici. V opačném případě robot z odpovědi ve formatu JSON získá informace o názvu knihy, autorovi a žánrech. Tyto informace budou robotem přehrány.

8. Zamíříme ke krabicím

Metoda `go_to_box_area`

Historie poloh robota byla uložena v proměnné position_history a aktualizována při každém pohybu robota. K návratu do původní polohy se použije první uložená poloha.

9. Hledáme správnou krabici

Metody `find_box`, `get_text_from_image`, `move_to_next_box`, `box_not_found_decorations`

Robot vyfotí krabici a odešle ji na podadresu */category*. Z odpovědi ve formatu JSON načte žánr. Pokud se žánr krabice objeví v žánrech knihy nebo je žánr knihy "Uncategorized", provede se další krok. V opačném případě se robot posune do strany o pevně danou vzdálenost a zopakuje předchozí algoritmus. Pokud server nenajde text na obrázku, robot oznámí, že krabici nenašel.

10. Krabice nalezena

Metody `box_found_decorations`, `go_to_position`, `wait_for_starting_touch`

Robot řekne informace o krabici a vrátí se do původní polohy. Zároveň upozorní, že pokud chce uživatel znovu spustit algoritmus, musí se dotknout hlavy robota.

Server

Server byl naprogramován v jazyce Python a pro obsluhu komunikace byl zvolen vestavěný *http.server*. Server má na starosti dva úkoly:

1. Rozpoznávání textu na krabicích

Při požadavku z url „\category“ se obrázek pořízený robotem nahraje na sever, kde je následně předzpracoován. Výběr metod pro předzpracování probíhal na základě nejlepšího zpracování testovcích dat bez fotografii z reálného prostředí (k nalezení v adresáří serveru) a následně korigován podle výsledkú při práci s robotem v laboratoři. Při výběru metod zpracování mi pomohly články [1] [2] a [3].

2. Rozpoznávání textu na obálce knihy

Při požadavku z url „/cover“ se obrázek pořízený robotem nahraje na server a snažíme se zjistit jméno knihy. V tomto kroku se předzpracování obrázku ukázalo spíše kontraproduktivní a tak jsme se rozhodli pouze o otáčení obrázku a využití balíčku nonsense (https://github.com/casics/nostril) který jisté chyby odhalí, dále se omezujeme na alespoň pětiznaková slova a nejpravděpodobnější dvojici poté posíláme jako dotaz na Google Books API . V případě úspěchu se knihu podle ISBN pokusíme dovyhledat v Open Library a získané informace v JSON formátu vrátit ke zpracování klientovi.

1https://towardsdatascience.com/pre-processing-in-ocr-fc231c6035a7
2https://nextgeninvent.com/blogs/7-steps-of-image-pre-processing-to-improve-ocr-using-python-2/
3https://pyimagesearch.com/2021/11/22/improving-ocr-results-with-basic-image-processing/